DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico I

Perceptrón multicapa

Redes Neuronales
Primer Cuatrimestre 2020

Integrante	LU	Correo electrónico
Ilan Olkies	250/17	ilanolkies@gmail.com
Joaquín Naftaly	816/17	joaquin.naftaly@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Introducción

En este trabajo evaluamos el aprendizaje supervisado con distintos modelos de perceptrones para dos problemas:

1. Problema de clasificación: predecir el diagnóstico final de cáncer de mamas.
2. Problema de regresión: predecir los valores de carga energética para la calefacción y refrigeración de edificios.

Para ambos problemas contamos con un dataset que nos permitirá construir el modelo y probar su efectividad con datos conocidos. El objetivo es encontrar el modelo que mejor ajuste a los datos. Los distintos modelos están dados por parámetros que describimos debajo.

En primer lugar, los datos utilizados para entrenar al modelo serán preprocesados para facilitar el entrenamiento y reducir el riesgo de caer en un mínimo local. Luego, se realiza el proceso de *entrenamiento*. El modelo aprende de instancias conocidas dadas para mejorar su precisión. Una vez entrenado, el modelo puede ser usado para evaluar nuevas instancias. Evaluamos un porcentaje del dataset que no fue utilizado en el entrenamiento para comparar los resultados obtenidos con los resultados esperados.

2. Modelo

El perceptrón multicapa es una red neuronal capaz de resolver problemas que no son linealmente separables. El modelo matemático representa los parámetros de las neuronas en forma de matriz, que se modifica iterativamente para adaptar a los datos conocidos lo mejor posible. Para esto, en cada iteración, el algoritmo calcula el error cometido en la aproximación de los datos de entrada y aplica correcciones para mejorar su rendimiento.

2.1. Activación

El perceptrón funciona con un modo de activación feed-forward. El conjunto de instancias a evaluar activa las neuronas de entrada, se activan las distintas capas ocultas sucesivamente, y finalmente las neuronas de salida. Cada neurona tiene una función de activación dada por la función tangente hiperbólica tanh, que tiene imagen $(-1, 1)$.

Uno de los paramétros que debemos determinar para mejorar nuestro modelo es el tamaño y la cantidad de capas ocultas. El tamaño de las capas de entrada y salida esta dado por los datos del problema.

2.2. Preprocesamiento

En ambos ejercicios los rangos de las variables de entrada son muy diversos y amplios (Ver Figura 1). Para disminuir la dificultad del entrenamiento debemos transformar los datos proporcionalmente para que tengan la misma escala. Para que esta escala funcione bien con la función de activación los datos deben estar en el rango $[-1, 1]$. Primero usamos la estrategia de *normalización* definida por:

$$x' = \frac{x - x_{\text{mín}}}{x_{\text{máx}} - x_{\text{mín}}} \tag{1}$$

Ahora $x' \in [0, 1]$. Luego, para que nuestros datos estén en el rango $[-1, 1]$ aplicamos:

$$x'' = 2 \times x' - 1 \tag{2}$$

A modo de ejemplo, los datos del problema 1 sin normalizar y normalizados se ven así:

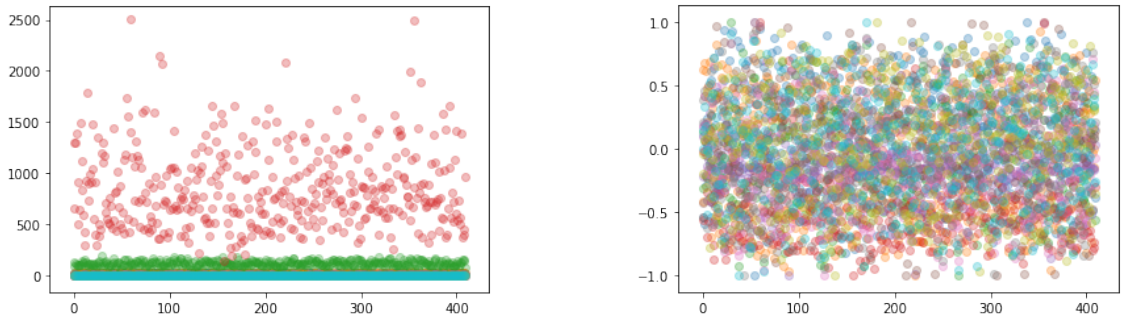


Figura 1: Datos del problema 1

Si observamos una única variable del problema, podemos ver que la proporción entre los valores se mantiene y la escala cambia.

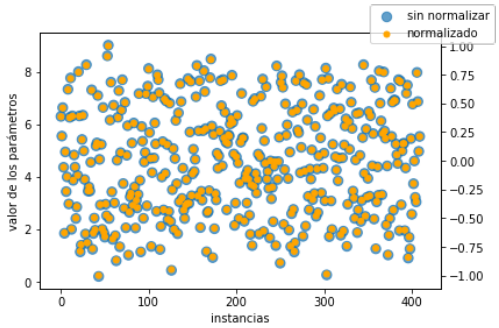


Figura 2: Datos de una sola variable del problema 1

2.3. Entrenamiento

El entrenamiento consta de tres etapas. La primera es la activación de las capas feed-forward. La segunda en la que con el error cometido en la primer etapa, comparándolo con el resultado esperado, se calculan las correcciones de los pesos de cada capa mediante *Backpropagation*. Y la tercera en la que se aplican esas correcciones a las matrices de pesos para lograr disminuir el error.

2.4. Mini-lotes

El entrenamiento se puede realizar procesando las instancias de a una o de a grupos o lotes. Además es conveniente procesar las instancias en un orden aleatorio para mejorar el entrenamiento. Esta estrategia se llama *mini-lotes*. El tamaño de estos lotes es otro parámetro que vamos a determinar para elegir nuestro modelo.

2.5. Propagación

En cada iteración del aprendizaje, se calcula el error cometido en la salida comparando con la respuesta esperada. Este error se propaga desde la ultima capa hacia las demás mediante el algoritmo de backpropagation en el que cada neurona recibe una proporción del error dependiendo que tanto haya contribuido para lograr ese error.

El calculo de las correcciones a realizar en las matrices de pesos también depende proporcionalmente de una variable llamada *learning rate* que permite aplicar los cambios de forma gradual. Este es otro parámetro que determinaremos para nuestro modelo.

2.6. Testing

Para evaluar la precisión del modelo, el conjunto de datos se divide en dos conjuntos: un subconjunto de entrenamiento o *training* y un subdonjunto de prueba o *testing*. Estos subconjuntos son disjuntos, al modelo se le enseñan los datos del conjunto de *testing* y se lo evalúa con otros datos que no son conocidos por el modelo, el conjunto de *testing*.

3. Ejercicios

Para encontrar el mejor modelo para los distintos problemas seguimos un procedimiento iterativo muy simple. Proponemos unos parámetros para el learning rate, cantidad y tamaño de capas ocultas, cantidad de epochs y cantidad de *mini-lotes*; entrenamos al perceptron con el 75 % de los datos y lo evaluamos con el otro 25 %. Luego ajustamos los parámetros para mejorar la precisión y repetimos.

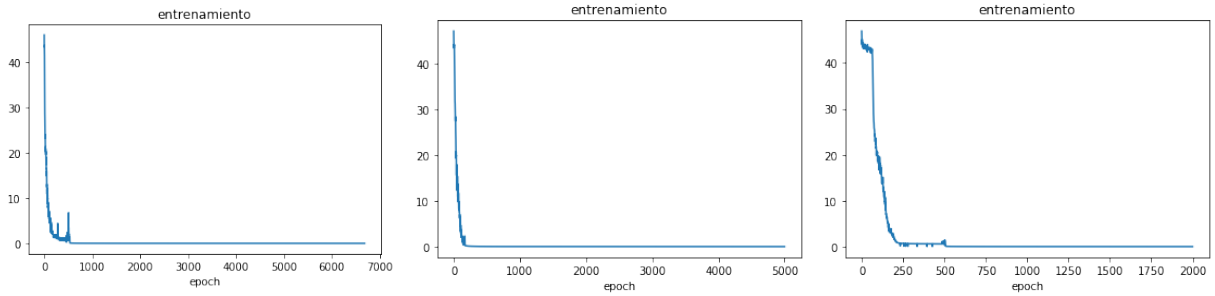
3.1. Diagnóstico de cáncer de mamas

El primer problema es un problema de clasificación. Usamos nuestro perceptrón para clasificar si el diagnostico final de cáncer de mamas es Benigno o Maligno basándose únicamente en un examen médico realizado que indica ciertas características del tumor. Para determinar esto consideramos que el resultado es Benigno si el valor de la neurona de salida es positivo, y Maligno si el valor es negativo. Además, consideramos que un modelo es mejor que otro si tiene un mayor *promedio* de acierto.

Dados los datos de los exámenes, nuestro modelo debe tener una capa de entrada de 10 entradas, una para cada propiedad del examen, y capa de salida con una única neurona que determina según su signo el resultado final de la clasificación.

La búsqueda del mejor modelo consta de variar los parámetros nombrados anteriormente de forma gradual, comparando el promedio de aciertos entre cada modelo. Nuestro modelo óptimo logra acertar un **94.1 %** de los tests usando dos capas ocultas de 10 neuronas, el *learning rate* en 0,02 y aplicando la metodología de *mini-lotes* con lotes de 7 instancias.

Durante la experimentación observamos que muchos de los modelos que son muy acertados no mejoran su estimación si la cantidad de épocas aumenta considerablemente o el error mínimo aceptado es menor. Esto se puede observar en la Figura 3 donde entrenamos nuestro modelo óptimo durante 2000, 5000 y 15000 épocas máximas, con una cota de error mínimo cometido. Comparando el entrenamiento de 2000 y 5000 iteraciones podemos observar que el error mejora pero el nivel de precisión es casi el mismo, mientras que para 15000 iteraciones máximas el modelo deja de entrenar por alcanzar la cota de error pero resulta en una peor precisión. Este suceso es conocido como *overfitting* y se da cuando el modelo se entrena más de lo necesario.



(a)
10000 iteraciones
error 0.001082...
prec. 0.8431372549019608

(b)
5000 iteraciones
error 0.001382...
prec. 0.9313725490196079

(c)
2000 iteraciones
error 0.003083...
prec. 0.941747572815534

Figura 3: Error cometido durante el entrenamiento del modelo

Observemos ahora los resultados obtenidos en sí. En la Figura 4 podemos ver el resultado de la estimación de nuestro conjunto de *testing* aplicado a nuestro modelo. En la primer imagen vemos el funcionamiento de la categorización por signo. La primer fila muestra el valor de activación de la neurona de salida, la segunda fila muestra si ese valor es positivo o negativo y la tercera el resultado esperado para un subconjunto de los resultados. Observamos que los valores grises no son del todo precisos al estimar. La segunda figura nos muestra la dispersión de los resultados, y la tercera muestra la dispersión solo para los resultados no acertados y los que su valor de activación esta en el intervalo $(-0{,}95, 0{,}95)$. Observamos que solo 2 de los 6 resultados fallidos están fuera de este rango, y solo 13 acertados dentro. Este resultado sugiere no considerar como resultados finales los resultados que están dentro del rango especificado.

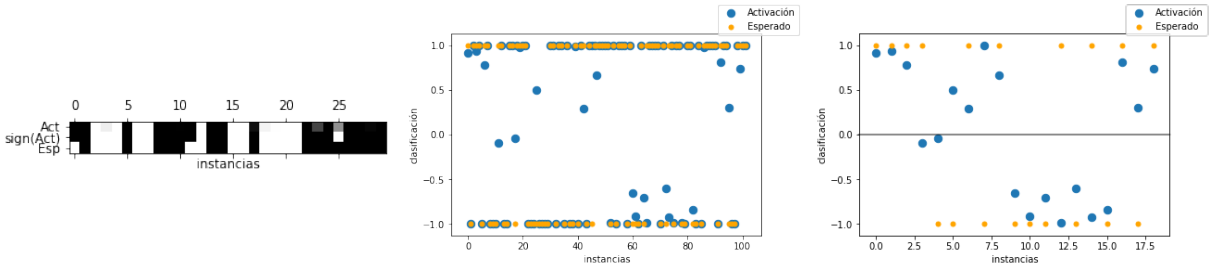


Figura 4: Resultado determinado por el modelo entrenado comparado con el resultado esperado

3.2. Eficiencia energética

Este es un problema de regresión. Dados los resultado del análisis de las edificaciones, se debe intentar aproximar el valor energético de calefacción y refrigeración necesarios. En este caso no debemos categorizar los resultados, entonces consideramos que un modelo es mejor que otro si aplicar el error promedio por instancia de la suma de las diferencias cuadradas entre la respuesta deseada y la obtenida para todas las unidades de salida es menor. El error esta en el rango $(0, 8)$.

Nuestro modelo tiene una capa de 8 neuronas de entrada y 2 neuronas de salida, la salida será comparada con los resultados esperados. Estos resultados esperados tienen que estar normalizados al igual que los datos de entrenamiento.

El mejor modelo encontrado para este caso tiene un **error de 0.0551 o de un 6.9 %** usando una única capa intermedia de 8 neuronas, *learning rate* de 0.02, y 5 lotes.

Veamos ahora en la Figura 5 los resultados del error cometido en el entrenamiento en cada época. Efectivamente nuestro perceptrón también funciona para este problema.

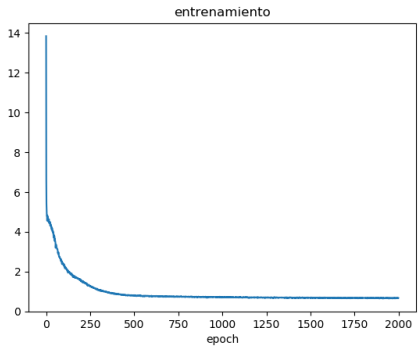


Figura 5: Datos de una sola variable del problema 1

Finalmente observemos la dispersión de los resultados. Podemos ver en la Figura 6 que una gran parte de los datos es estimada correctamente, pocos valores se encuentran distantes a su valor esperado. En la

tercer imagen de izquierda a derecha podemos ver esta comparación para 20 de las instancias testeadas con el valor respectivo del error cometido en la estimación de cada instancia.

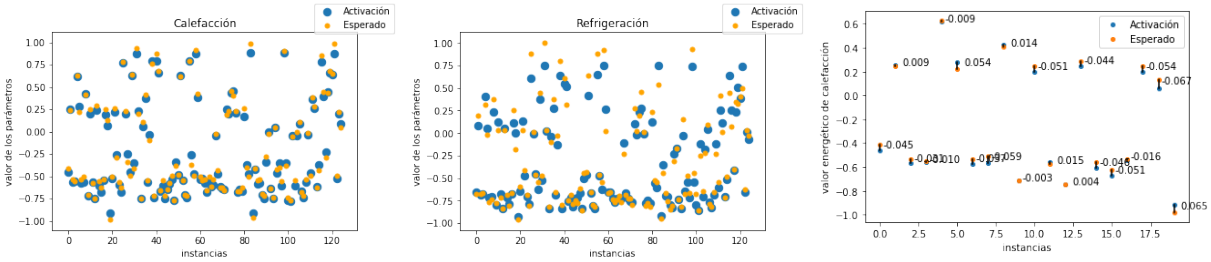


Figura 6: Resultado determinado por el modelo entrenado comparado con el resultado esperado