Redes Neuronales Multicapa para Regresion y Clasificacion

Marceca, Gino

15 de junio de 2018

1. Introducción

En este trabajo se presenta el estudio de redes neuronales multicapa aplicadas a problemas de clasificación y regresión. El primero corresponde al diagnóstico de cáncer de mamas, cuyo entrenamiento se realizo en un subconjunto representativo tomado al azar de 409 datos con 10 características cada uno provenientes de imágenes digitalizadas de muestras reales de células. Junto con estas características se encuentra también el diagnóstico final, determinado junto con otras pruebas en donde se indica si la muestra analizada pertenecía a un tumor maligno o benigno.

Para el problema con regresión se determino la carga energética óptima para calefaccionar y refrigerar edificios en función de ciertas características de los mismos. El conjunto de datos contiene 8 atributos y 2 respuestas medidos en edificios de distintas características. El entrenamiento se realizo en un subconjunto tomado al azar de 500 datos.

Para validación del funcionamiento de la red, se emulo la lógica XOR para el problema de clasificación. Para validar el de regresión, se generaron par de numeros aleatorios (x, y) con $x = U[0, 1]$ e $y = \sin(2\pi x) + N(0, 1)$ y se obtuvieron los valores predichos de $\hat{y}$ usando como características de entrada de la red x^j con $j = (1, 2, \ldots, M)$.

Una vez que la validación fue exitosa, se prosiguio a resolver los problemas planteados. El modelo se programo en el lenguaje *Python* versión 3.5.4.

2. Secuencia del Programa

Se utilizó un mismo modelo de red tanto para el problema de clasificación como el de regresión. A continuación se detalla en formato de pseudocódigo el algoritmo desarrollado.

2.1. Modelo

El modelo utilizado fue una red neuronal *feedforward* de L capas cuya secuencia consiste en:

1. Se inicializan los parámetros de pesos W y sesgos b de la red para todas las L capas.

```
Loop i=1 hasta L:
W^(i) = N(0, σ_init)
      b^(i) = 0
      end

W^(l)  -- matriz de peso de dimension l × (l − 1)
b^(l)  -- vector de sesgos de dimension l × 1
l  -- cantidad de nodos en la capa l-esima
```

2. Se procede a un loop donde se actualizan los pesos en forma iterativa usando *stochastic gradient descent* (SGD).

```
Loop in iterations:

    (a) Propagacion hacia adelante (FP)

    (b) Computo de la funcion de error

    (c) Propagacion hacia atras (BP)

    (d) Actualizacion de los parametros usando SGD

    end
```

(a) Propagación hacia adelante (FP)

Los valores de cada nodo en la capa l se calculan a partir de los anteriores mediante la siguiente expresión:

$$Z^{(l)} = W^{(l)} A^t_{(l-1)} + b^{(l)}$$

El valor de los nodos estan dados por la función de activación $\sigma(Z)$ en dicha capa evaluada en $Z^{(l)}$:

$$A_{(l)} = \sigma(Z^{(l)})$$

Donde $\sigma(Z)$ es la función *sigmoidea* dada por: $\sigma(Z) = \frac{1}{1+e^{-Z}}$

Se calculan estos valores en cada capa hasta llegar a la útima. Esta consta de un único nodo evaluado en la función de activación $h(Z)$, el cual indica el valor predicho $\hat{y}$ para el set de datos de entrada x.

$$\hat{y} = h(Z^{(L)})$$

Para el problema de clasificación se utilizó $h(Z) = \sigma(Z)$ y para el de regresión $h(Z) = Z$

(b) Cómputo de la función de error

Una vez obtenidos los valores predichos $\hat{y}$, se calcula el error $\mathcal{L}(y, \hat{y})$ cometido en la predicción.
Para el problema de clasificación se utilizó el error de regresion logística dado por

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{m} \sum_i^m \left(y^{(i)} log(\hat{y}^{(i)}) + (1 - y^{(i)}) log(1 - \hat{y}^{(i)}) \right) \tag{1}$$

Para el problema de regresión se utilizó el error cuadrático medio:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{2m} \sum_i^m \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \tag{2}$$

donde m es la cantidad de datos para los sets de entrenamiento o de prueba.

(c) Propagacion hacia atras (BP)

Para poder efectuar la actualización de los parámetros con SGD no solo se necesita de la función error, sino tambien de las derivadas respecto a dichos parámetros. Para poder calcular las derivadas, se utilizo el procedimiento de BP. La derivada del error respecto a una variable s ($\frac{d\mathcal{L}}{ds}$) se denota simplemente como ds.

Se comienza calculando $d\hat{y}$ y luego se prosigue, de adelante hacia atras, a calcular el resto de las derivadas.

Derivando las ecuaciones 1 y 2 respecto de $\hat{y}$ se obtiene que:

$$d\hat{y}_{\text{clasificacion}} = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}})$$
$$d\hat{y}_{\text{regresion}} = \hat{y} - y$$

Las expresiones que resultan de propagar las derivadas son:

$$dW^{(l)} = dZ^{(l)} A^{(l-1)T}$$
$$db^{(l)} = \sum_i^m dZ^{(l)(i)}$$
$$dA^{(l-1)} = W^{(l)T} dZ^{(l)}$$

donde $dZ^{(l)} = \frac{d\mathcal{L}}{dZ^{(l)}} = \frac{d\mathcal{L}}{dA^{(l)}} \frac{dA^{(l)}}{dZ^{(l)}}$

Para el problema de clasificación se tiene que $\frac{dA^{(l)}}{dZ^{(l)}} = \sigma'(Z) = \sigma \times (1 - \sigma)$ para $l = (1, ..., L)$.

Para el de regresión se tiene que $\frac{dA^{(l)}}{dZ^{(l)}} = \sigma \times (1 - \sigma)$ para $l = (1, ..., L-1)$ y $\frac{dA^{(l)}}{dZ^{(l)}} = 1$ para $l = L$

(d) Actualización de los parámetros

Teniendo $\mathcal{L}$ y sus derivadas, se actualizan los parámetros de la siguiente manera:

$$W^{(l)}_{\text{new}} = W^{(l)}_{\text{old}} - \alpha \frac{d\mathcal{L}}{dW^{(l)}_{\text{old}}}$$
$$b^{(l)}_{\text{new}} = b^{(l)}_{\text{old}} - \alpha \frac{d\mathcal{L}}{db^{(l)}_{\text{old}}}$$

donde α es un parámetro conocido como *rate* de aprendizaje.

El proceso vuelve a comenzar por (a) y continua hasta conseguir un valor de pesos suficientemente cercanos al óptimo.

3. Validación del Programa

Antes de proseguir con los resultados finales, se describe el proceso de validación de la red.

3.1. Clasificación: Lógico XOR

Para emular un lógico XOR, se utilizo el modelo de red mas simple posible para este problema, el cuál consiste de una capa oculta con dos nodos. Las funciones de activación usadas fueron las sigmoideas. Los inputs de entrada X fue una matriz de dimensión 2×4 dada por:

$$X = \begin{Bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{Bmatrix}$$

Los valores y de entrenamiento fueron los correspondientes a un XOR:

$$y = \begin{Bmatrix} 0 & 1 & 1 & 0 \end{Bmatrix}$$

Los hyper-parámetros de la red utilizados que resultaron en convergencia fueron $\alpha = 0{,}5$, #hidden_layers=1, #hidden_units=2, función de activación: sigmoidea.

Luego de 3000 iteraciones el resultado predicho fue $\hat{y} = (0{,}014\ 0{,}98\ 0{,}99\ 0{,}012)$ que se acerca mucho al deseado.

Un parámetro que fue importante para la convergencia fue la elección de σ para la inicialización de la matriz de pesos W. Se observó convergencia solo para valores $\sigma_{\text{init}} \gtrsim 0{,}5$.

3.2. Regresión: Fit Polinomial de una Sinusoide

Se generaron par de numeros aleatorios (x, y) con $x = U[0, 1]$ e $y = \sin(2\pi x)$ con ruido gaussiano $N(0, 1)$. Usando como caracteristicas de entrada de la red:

$$X = \begin{Bmatrix} x_1 & x_1^2 & \dots & x_1^M \\ \vdots & x_2^2 & \dots & x_2^M \\ x_N & x_N^2 & \dots & x_N^M \end{Bmatrix}$$

con N el numero de datos generados y M el grado del polinomio.

En la Figura 1 se observan las curvas de aprendizaje para los sets de entrenamiento y de pruba (izquierda) y los valores generados y y los predichos $\hat{y}$ en función de x (derecha). Como se puede ver en las curvas de aprendizaje, no se observa *overfitting* ni *underfitting*. Asi mismo, los valores predichos se acercan a los puntos (rojos) verdaderos (sin ruido) que genero el set de entrenamiento (verde).

Los hyper-parámetros utilizados en esta red fueron $\alpha = 0{,}3$, #hidden_layers=1, #hidden_units=7, función de activación: sigmoidea, #iteraciones = 1500, $\sigma_{\text{init}} = 0{,}5$.

La cantidad de datos generados fue de $N = 100$ y el grado del polinomio utilizado fue $M = 6$.

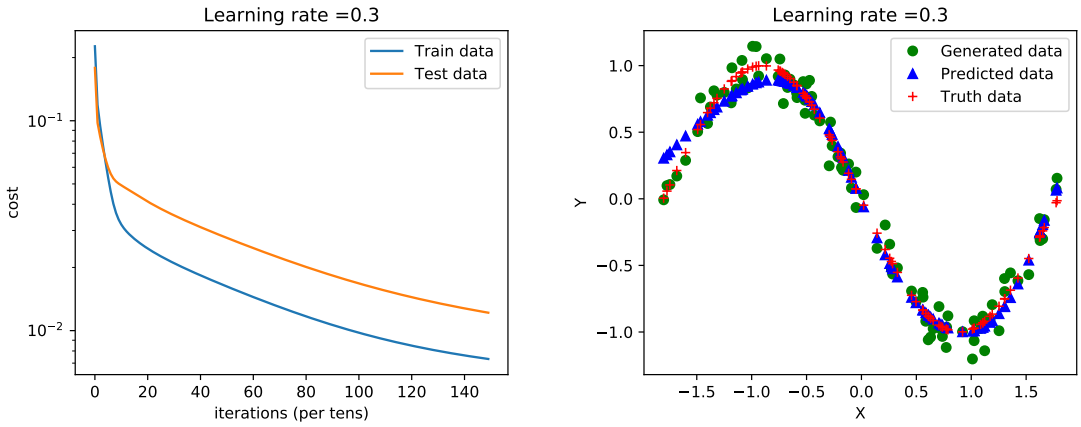

Figura 1: Curvas de aprendizaje (izquierda) para los sets de entrenamiento y prueba. Valores generados y predichos y ($\hat{y}$) en función de x (derecha)

4. Resultados

4.1. Clasificación

4.1.1. Validación de la Red

Para encontrar una configuración en los hyperprámetros optima de la red se dividio el set de datos en entrenamiento (60 %), validación (20 %) y testeo (20 %). Distintas configuraciones fueron entrenadas y validadas en los sets respectivos. En la Figura 2 se presentan las curvas de aprendizaje para distintas configuraciones utilizadas. Los hyperparametros seleccionados para la validación fueron el *learning rate* (λ), numero de nodos en la capa oculta (n_h) y numero de capas ocultas (h_L). Como es de esperarse, la función de costo decrece mas lentamente a medida que disminuye λ, como se observa en la figura de la izquierda. Por otro lado, al aumentar el numero de nodos y/o numero de layers, aumenta el nivel de *overfitting*, como se ve en la figura de la derecha. La configuración optima basada en el set de validación fue:
$\lambda = 1{,}0$, #hidden_layers=1, #hidden_units=10, #iteraciones = 3000, función de activación: sigmoidea, $\sigma_{\text{init}} = 0{,}2$.
Esta fue la que se utilizo para evaluar la performance final en el set de testeo.

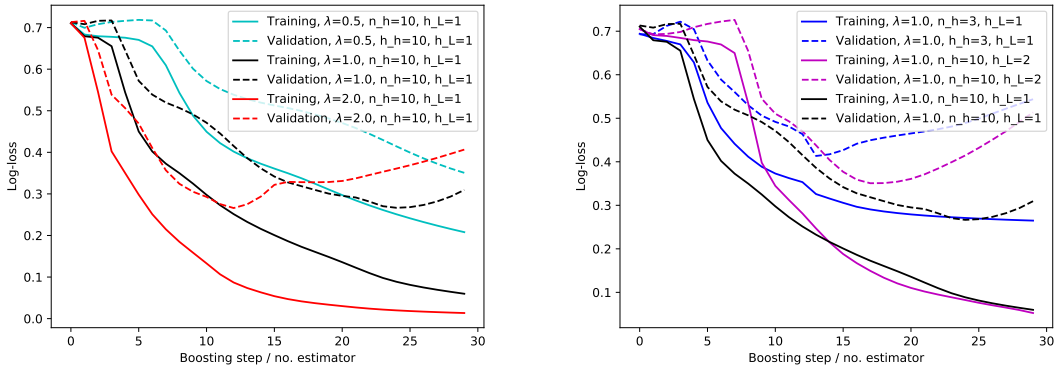

Figura 2: Curvas de aprendizaje para distintas configuraciones del *learning rate* (λ) (izquierda) y para distintas configuraciones en el numero de nodos en la capa oculta (n_h) y el numero de capas ocultas (h_L) (derecha)

4.1.2. Evaluación (perfomance final)

En la Figura 3 se observan las curvas de aprendizaje obtenidas para el set de entrenamiento y testeo (izquierda), y la variable predicha $\hat{y}$ (derecha) para los eventos de tumores malignos y benignos.

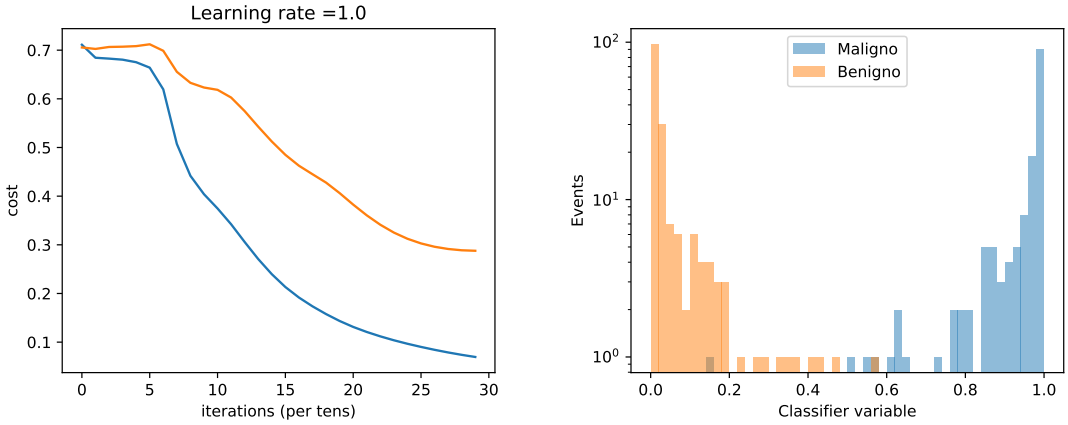

Figura 3: Curvas de aprendizaje (izquierda) para los sets de entrenamiento y prueba. Variable clasificadora resultante (derecha)

La performance se evaluo en el set de testeo clasificando los tumores como malignos (benignos) si $\hat{y} > 0{,}5$ ($\hat{y} <= 0{,}5$) respectivamente. La eficiencia obtenida fue de 86,6 %.

4.2. Regresión

4.2.1. Validación de la Red

Al igual que en el caso de clasificación, se estudio la performance para distintas configuraciones del espacio de hyperparametros. En este caso se utilizo el set de testeo como validación por motivos de baja estadística, lo que impedia separar en tres regiones estadísticamente representativas.
En la Figura 4 se presentan las curvas de aprendizaje para distintas configuraciones utilizadas. Los hyperparametros seleccionados para la validación fueron el *learning rate* (λ), numero de nodos en la capa oculta (n_h) y numero de capas ocultas (h_L). Se obtuvieron las mismas concluciones que en el caso de clasificación. La configuración optima para el problema de calefacción basada en el set de testeo fue: $\lambda = 0{,}01$, #hidden_layers=2, #hidden_units_1L=7, #hidden_units_2L=5, #iteraciones = 60000, función de activación: linear, $\sigma_{\text{init}} = 0{,}3$.

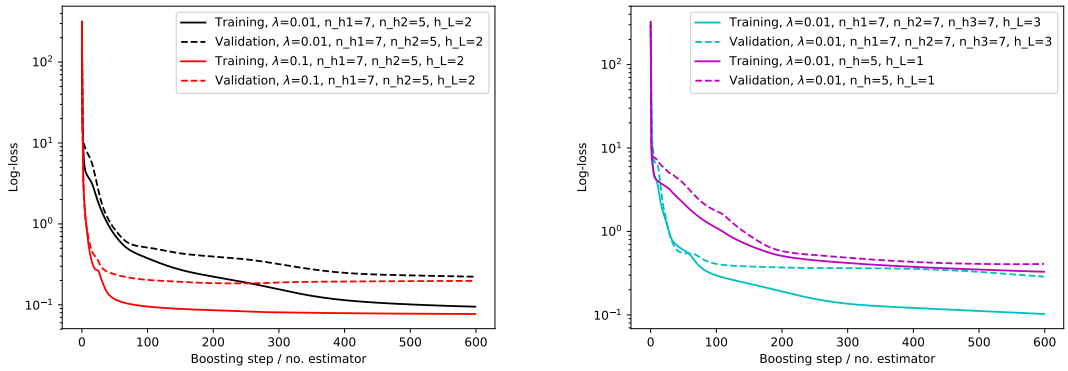

Figura 4: Curvas de aprendizaje para distintas configuraciones del *learning rate* (λ) (izquierda) y para distintas configuraciones en el numero de nodos en la capa oculta (n_h) y el numero de capas ocultas (h_L) (derecha)

Un estudio similar se realizo para el problema de refrigeración. Los hypeparametros optimos encontrados fueron:
$\lambda = 0{,}1$, #hidden_layers=2, #hidden_units_1L=7, #hidden_units_2L=2, #iteraciones = 20000, función de activación: linear, $\sigma_{\text{init}} = 0{,}6$.

4.2.2. Evaluación (perfomance final)

En la Figura 5 se observan las curvas de aprendizaje para los sets de entrenamiento y prueba para los dataset de calefacción y refigeación (izquierda y derecha respectivamente).

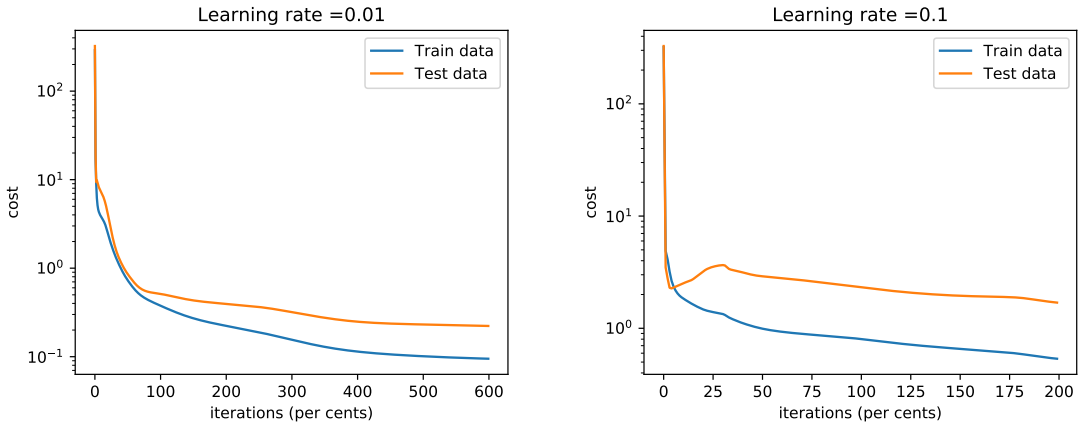

Figura 5: Curvas de aprendizaje para los sets de entrenamiento y prueba para los dataset de calefacción y refigeación (izquierda y derecha respectivamente)

5. Conclusiones

En este trabajo se estudió el uso de redes neuronales multicapa como algoritmo de aprendizaje para la clasificación de tumores de cáncer de mamas y la predicción de la energía necesaria para calefaccionar y refrigerar una casa, basado en un set de datos supervisado. Se programo la misma paso a paso y se valido el programa en un lógico XOR para el problema de clasificación, y en la predicción de un set de datos generados por una sinusoide con ruido gaussiano para el caso de regresión. Para la validación de la red se utilizo un set de validación en el cual se probaron distintas configuraciones de hyperparametros. Los parametros optimos encontrados fueron:

Clasificación:

$\lambda = 1{,}0$, #hidden_layers=1, #hidden_units=10, #iteraciones = 3000, función de activación: sigmoidea, $\sigma_{\text{init}} = 0{,}2$

Regresión-calefacción:

$\lambda = 0{,}01$, #hidden_layers=2, #hidden_units_1L=7, #hidden_units_2L=5, #iteraciones = 60000, función de activación: linear, $\sigma_{\text{init}} = 0{,}3$.

Regresión-refrigeración:

$\lambda = 0{,}1$, #hidden_layers=2, #hidden_units_1L=7, #hidden_units_2L=2, #iteraciones = 20000, función de activación: linear, $\sigma_{\text{init}} = 0{,}6$.

La performance final obtenida en el set de prueba para el problema de cáncer de mamas fue de 86,6 %. Las cuervas de aprendizaje muestran en ambos casos convergencia tanto en el set de entrenamiento como el de prueba para el problema de regresión y clasificación.