Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Especialidad: Computación
Cuarto curso. Primer cuatrimestre

Introducción a los modelos computacionales.

Práctica 1: Implementación del perceptrón multicapa.

Antonio Ariza García
DNI
i62argaa@uco.es

Curso académico 2020-2021
Córdoba, 27 de febrero de 2021

Índice

Índice de tablas

List of Algorithms

1. Introduccion a los modelos de redes neuronales

Las redes neuronales son un modelo computacional basado en el funcionamiento de las neuronas. Consiste en un conjunto de unidades llamadas neuronas, conectadas entre sí para trasmitir señales. La información de entrada atraviesa la red neuronal produciendo unos valores de salida. Este trabajo consiste en implementar el algoritmo de retropropagación para entrenar un perceptrón multicapa. Para ello se desarrollará un programa capaz de realizar este entrenamiento, con distintas posibilidades en cuanto a la parametrización del mismo.

La arquitectura de la red (número de capas y número de nodos en cada capa), junto con la tipología de los nodos (tipos de funciones de activación a considerar), son parámetros decisivos del algoritmo que hay que buscar por prueba y error o por validación cruzada.

El sobre-entrenamiento en redes neuronales suele venir provocado por dos causas: Entrenamiento demasiado largo (condición de parada inadecuada). Redes demasiado complejas (muchas neuronas o muchas capas).

2. Implementación

2.1. Algoritmo de retropropagación online

Algorithm 1 Retropropagación online

```
1: PesosAleatorios()
2: for all patron con entrada x y salida d do
3:     Δw_ji^h ← 0
4:     forwardPropagation()
5:     backPropagation()
6:     accumulateChange()
7:     weightAjustment()
8: end for
```

Algorithm 2 forwardPropagation

for all capa i de 1 a H do **do**
  for all neurona j de la capa i do **do**
    $SigmoidSum \leftarrow w^i_{j0}$
    for all neurona k de la capa i-1 do **do**
      $SigmoidSum \leftarrow SigmoidSum + x^{i-1}_k * x^i_{jk-1}$
    end for
  end for
end for

Algorithm 3 accumulateChange

$\eta \leftarrow dEta$
for all capa i de 1 a H do **do**
  for all neurona j de la capa i do **do**
    for all neurona k de la capa i-1 do **do**
      $ultimo\Delta w^i_{jk} \leftarrow \Delta w^i_{jk}$
      $\Delta w^i_{jk} \leftarrow \Delta w^i_{jk} + dX^i_j * X^{i-1}_{k-1}$
    end for
    $ultimo\Delta w^i_{j0} \leftarrow \Delta w^i_{j0}$
    $\Delta w^i_{j0} \leftarrow \Delta w^i_{j0} + dX^i_j$
  end for
end for

Algorithm 4 weightAdjustment

$\eta \leftarrow dEta$
for all capa i de 1 a H do **do**
  for all neurona j de la capa i do **do**
    for all neurona k de la capa i-1 do **do**
      $w^i_{jk} \leftarrow w^i_{jk} - \eta * \Delta w^i_{jk} - \mu * \eta * ultimo\Delta w^i_k jk$
    end for
    $w^i_{j0} \leftarrow w^i_{j0} - \eta * \Delta w^i_{j0} - \mu * \eta * ultimo\Delta w^i_{j0}$
  end for
  $\eta \leftarrow \eta dDecremento^{-(nofLayer-i)}$
end for

3. Resultados Prácticos

Probaremos distintas configuraciones de la red neuronal y ejecutaremos cada configuración con cinco semillas (1, 2, 3, 4 y 5). A partir de los resultados obtenidos, se obtendrá la media y la desviación típica del error. Se deberá calcular el error MSE para el conjunto de entrenamiento y el error MSE para el conjunto de test.

Para valorar como funciona el algoritmo implementado en esta práctica, emplearemos un total de cuatro bases de datos:

- Problema XOR: esta base de datos representa el problema de clasificación no lineal del XOR. Se utilizará el mismo fichero para train y para test.
- Función seno: esta base de datos está compuesta por 120 patrones de train y 41 patrones de test. Ha sido obtenida a˜nadiendo cierto ruido aleatorio a la función seno
- Base de datos quake: esta base de datos está compuesta por 1633 patrones de train y 546 patrones de test. Se corresponde con una base de datos en la que el objetivo es averiguar la fuerza de un terremoto (medida en escala sismológica de Richter). Como variables de entrada, utilizamos la profundidad focal, la latitud en la que se produce y la longitud.
- Base de datos parkinsons: esta base de datos está compuesta por 4406 patrones de train y 1469 patrones de test. Contiene, como entradas o variables independientes, una serie de datos clínicos de pacientes con la enfermedad de Parkinson y datos de medidas biométricas de la voz, y, como salidas o variables dependientes, el valor motor y total del UPDRS.

3.1. Descripción de los distintos parámetros

A continuación se explicarán cada parámetro que podrá recibir el programa:

- Argumento t: Indica el nombre del fichero que contiene los datos de entrenamiento a utilizar. Sin este argumento, el programa no puede funcionar.

- Argumento T: Indica el nombre del fichero que contiene los datos de test a utilizar. Si no se especifica este argumento, utilizar los datos de entrenamiento como test.
- Argumento i: Indica el número de iteraciones del bucle externo a realizar. Si no se especifica, utilizar 1000 iteraciones.
- Argumento l: Indica el número de capas ocultas del modelo de red neuronal. Si no se especifica, utilizar 1 capa oculta.
- Argumento h: Indica el número de neuronas a introducir en cada una de las capas ocultas. Si no se especifica, utilizar 5 neuronas.
- Argumento e: Indica el valor del parámetro eta (η). Por defecto, utilizar $\eta = 0;1$.
- Argumento m: Indica el valor del parámetro mu (μ). Por defecto, utilizar $\mu = 0;9$.
- Argumento v: Indica el ratio de patrones de entrenamiento a utilizar como patrones de validación. Por defecto, utilizar v = 0;0.
- Argumento d: Indica el valor del factor de decremento a utilizar por cada una de las capas. Por defecto, utilizar F = 1.
- Argumento w: Indica el nombre del fichero en el que se almacenarán la configuración y el valor de los pesos del modelo entrenado.

3.2. Prueba de distintas arquitecturas

Para esta primera parte, utilizaremos factor de momento, no utilizaremos conjunto de validación (v = 0;0) y no emplearemos decremento de la tasa de aprendizaje (F = 1). Se deberá probar un total de 12 arquitecturas:

- Con una capa oculta: {n:2:k}, {n:4:k}, {n:8:k}, {n:32:k}, {n:64:k}, {n:100:k}
- Con dos capas ocultas: {n:2:2:k}, {n:4:4:k}, {n:8:8:k}, {n:32:32:k}, {n:64:64:k}, {n:100:100:k}

Una vez decidida la mejor arquitectura para cada problema, probaremos todas las combinaciones de estos dos parámetros: v 2 f0;0; 0;15; 0;25g y F 2 f1; 2g.

Para la BD XOR el train y el test no varía, el mejor resultado sería el formado por dos capas ocultas con 100 neuronas cada una.

Arquitecturas	**Train error**	σ**(Train error)**	**Test error**	σ**(Test error**
{n:2:k}	0.163856	0.0679974	0.163856	0.0679974
{n:4:k}	0.114261	0.0503647	0.114261	0.0503647
{n:8:k}	0.072993	0.0451338	0.072993	0.0451338
{n:32:k}	0.014523	0.0014708	0.014523	0.0014708
{n:64:k}	0.009289	0.0014194	0.009289	0.0014194
{n:100:k}	0.056003	0.108989	0.056003	0.1089892
{n:2:2:k}	0.248265	0.0025696	0.248265	0.0025696
{n:4:4:k}	0.24818	0.0021734	0.24818	0.0021734
{n:8:8:k}	0.187008	0.0612925	0.187008	0.0612925
{n:32:32:k}	0.007945	0.0014348	0.007945	0.0014348
{n:64:64:k}	0.002926	0.0001425	0.002926	0.0001425
{n:100:100:k}	0.001701	0.0001286	0.001701	0.0001286

Tabla 1: Errores Base de datos XOR en distintas arquitecturas

Para la BD Seno la mejor arquitectura sería con dos capas ocultas de 64 neuronas cada una, porque aunque la formada por dos capas ocultas y 100 neuronas tenga un error muy parecido, no es así para la desviación típica, pues presenta 5 veces más por lo que puede dar errores mayores.

Arquitecturas	**Train error**	σ**(Train error)**	**Test error**	σ**(Test error)**
{n:2:k}	0.0297282	3.08475e-05	0.0365526	0.000360624
{n:4:k}	0.029529	0.000411965	0.0363783	0.000223319
{n:8:k}	0.0294088	0.000534317	0.03617	0.000366612
{n:32:k}	0.0290472	0.000360613	0.0360009	0.000806839
{n:64:k}	0.0280052	0.000322043	0.0352727	0.000584671
{n:100:k}	0.0285983	0.000986158	0.036288	0.000428383
{n:2:2:k}	0.0297237	4.24326e-05	0.036248	0.000170634
{n:4:4:k}	0.0297821	3.06686e-05	0.0362179	0.000150539
{n:8:8:k}	0.0299407	0.000118453	0.036485	0.000630194
{n:32:32:k}	0.0296178	0.00064876	0.0361732	0.000585751
{n:64:64:k}	0.0269406	0.00080941	0.034039	0.000834419
{n:100:100:k}	0.0259549	0.00480641	0.0340658	0.004816921

Tabla 2: Errores Base de datos Seno en distintas arquitecturas

En el caso de Quake tenemos 32 neuronas y con una sola capa oculta.

Arquitecturas	Train error	σ(Train error)	Test error	σ(Test error)
{n:2:k}	0.0301022	7.06987e-05	0.027198	5.85502e-05
{n:4:k}	0.029967	5.33303e-05	0.0270686	8.04283e-05
{n:8:k}	0.029916	9.88792e-05	0.027052	7.48808e-05
{n:32:k}	0.0297653	2.91166e-05	0.026971	2.59938e-05
{n:64:k}	0.0297662	3.47861e-05	0.0270388	3.04479e-05
{n:100:k}	0.0297446	5.52488e-05	0.0270129	3.55988e-05
{n:2:2:k}	0.030148	4.05299e-05	0.0272508	4.38807e-05
{n:4:4:k}	0.0301119	3.07848e-05	0.0272228	3.27004e-05
{n:8:8:k}	0.0300999	7.20564e-05	0.0271879	8.2355e-05
{n:32:32:k}	0.0298273	8.00825e-05	0.0269784	4.39127e-05
{n:64:64:k}	0.0295095	7.63962e-05	0.0270458	5.78271e-05
{n:100:100:k}	0.0294047	1.42461e-05	0.0271243	1.80462e-05

Tabla 3: Errores Base de datos Quake en distintas arquitecturas

Para BD Parkinsons la mejor arquitectura está compuesta por dos capas ocultas y 64 neuronas en cada una.

Arquitecturas	Train error	σ(Train error)	Test error)	σ(Test error
{n:2:k}	0.0343992	9.38514e-06	0.0371213	9.13403e-05
{n:4:k}	0.0257571	0.000242337	0.0255326	8.04128e-05
{n:8:k}	0.0207401	0.00149025	0.0216091	0.0015686
{n:32:k}	0.0154767	0.000518124	0.0173277	0.000980353
{n:64:k}	0.0147414	0.000632241	0.0164462	0.000540946
{n:100:k}	0.0133494	0.000380026	0.0155019	0.000628582
{n:2:2:k}	0.0315812	0.000894328	0.032942	0.000926471
{n:4:4:k}	0.0234318	0.0014714	0.0239546	0.00123368
{n:8:8:k}	0.0146101	0.00204368	0.0154939	0.00246286
{n:32:32:k}	0.0090358	0.00131826	0.0112496	0.00153763
{n:64:64:k}	0.00777673	0.00121271	0.0101671	0.00159433
{n:100:100:k}	0.00878881	0.00585147	0.010591	0.00506543

Tabla 4: Errores Base de datos Parkinsons en distintas arquitecturas

4. Pruebas modificando el Factor de decremento

Una vez elegida la mejor arquitectura para cada una de las bases de datos, probaremos modificando el factor de decremento con valores {1,2}.

Para la mejor arquitectura de la base de datos XOR.

Arquitecturas	**F**	**Train error**	σ**(Train error)**	**Test error**	σ**(Test error)**
{n:100:100:k}	1	0.0017019	0.00012862	0.0017019	0.00012862
{n:100:100:k}	2	0.0145757	0.00814214	0.0145757	0.00814214

Tabla 5: Variaciones Factor decremento para base datos XOR

Para la mejor arquitectura de la base de datos Seno.

Arquitecturas	**F**	**Train error**	σ**(Train error)**	**Test error**	σ**(Test error)**
{n:64:64:k}	1	0.0269406	0.00080941	0.0340395	0.00083442
{n:64:64:k}	2	0.0284589	0.00039619	0.0360283	0.00110174

Tabla 6: Variaciones Factor decremento para base datos Seno

Para la mejor arquitectura de la base de datos Quake.

Arquitecturas	**F**	**Train error**	σ**(Train error)**	**Test error**	σ**(Test error)**
{n:32:k}	1	0.0297653	2.91166e-05	0.026971	2.59938e-05
{n:32:k}	2	0.0298116	2.73964e-05	0.0269728	2.02602e-05

Tabla 7: Variaciones Factor decremento para base datos Quake

Para la mejor arquitectura de la base de datos Parkinson.

Arquitecturas	**F**	**Train error**	σ**(Train error)**	**Test error**	σ**(Test error)**
{n:64:64:k}	1	0.0077767	0.00121271	0.0101671	0.00159433
{n:64:64:k}	2	0.0130969	0.0015374	0.0130969	0.00153748

Tabla 8: Variaciones Factor decremento para base datos Parkinsons

Aquí podemos comprobar como el factor de decremento apenas cambia los resultados, pero empeora nuestro modelo en todas las bases de datos.