Análisis con BackPropagation

1. Primeras pruebas

En primer lugar, se elabora el script con el que se realizarán las pruebas, en él se leen los datos de la base de datos ya preprocesada y se elaboran los datasets de entrenamiento, validación y test, para posteriormente construir la red neuronal, entrenarla, validarla y calcular los datos necesarios para mostrar porcentajes de acierto.

#Imports

from \_\_future\_\_ import division

from pybrain.datasets import SupervisedDataSet

from pybrain.supervised.trainers import BackpropTrainer

from pybrain.tools.shortcuts import buildNetwork

from pybrain.utilities import percentError

import numpy as np

import pylab as pl

import math as ma

#Leer las bases de datos

patternTrain = np.loadtxt("BreastCancerPreprocessedTrain.csv", dtype=float, delimiter=',')

patternValid = np.loadtxt("BreastCancerPreprocessedValid.csv", dtype=float, delimiter=',')

patternTest = np.loadtxt("BreastCancerPreprocessedTest.csv", dtype=float, delimiter=',')

#Conseguir el numero de filas y columnas

numPatTrain, numColsTrain = patternTrain.shape

numPatValid, numColsValid = patternValid.shape

numPatTest, numColsTest = patternTest.shape

#Generar el input

patternTrainInput = patternTrain[:, 1:numColsTrain]

patternValidInput = patternValid[:, 1:numColsValid]

patternTestInput = patternTest[:, 1:numColsTest]

#Generar salidas deseadas

patternTrainTarget = np.zeros([numPatTrain, 2])

patternValidTarget = np.zeros([numPatValid, 2])

patternTestTarget = np.zeros([numPatTest, 2])

#Crear los dataset supervisados

trainDS = SupervisedDataSet(numColsTrain-1, 2)

for i in range(numPatTrain):

patternTrainTarget[i, patternTrain[i, 0]] = 1.0

trainDS.addSample(patternTrainInput[i], patternTrainTarget[i])

validDS = SupervisedDataSet(numColsValid-1, 2)

for i in range(numPatValid):

patternValidTarget[i, patternValid[i, 0]] = 1.0

validDS.addSample(patternValidInput[i], patternValidTarget[i])

testDS = SupervisedDataSet(numColsTest-1, 2)

for i in range(numPatTest):

patternTestTarget[i, patternTest[i, 0]] = 1.0

testDS.addSample(patternTestInput[i], patternTestTarget[i])

#Crear red con una capa oculta

numHiddenNodes = 25

myLearningRate = 0.0001

myMomentum = 0.1

net = buildNetwork(numColsTrain-1, numHiddenNodes, 2, bias=True)

#Crear el trainer y hacer enternar el DS

trainer = BackpropTrainer(net, trainDS, learningrate=myLearningRate, momentum=myMomentum)

trainError = trainer.trainUntilConvergence(verbose=True, trainingData=trainDS, validationData=validDS)

#Crear la gráfica con los errores de validación y entrenamiento

pl.plot(trainError[0], label='Train Error')

pl.plot(trainError[1], label='Valid Error')

pl.xlabel('Epoch num')

pl.ylabel('Error')

pl.legend(loc='upper right')

pl.show()

#Obtener porcentajes

results = net.activateOnDataset(validDS)

patResult = -1

positivo = 0

negativo = 0

falsoPositivo = 0

falsoNegativo = 0

for i in range(numPatValid):

if max(results[i]) == results[i, 0]:

patResult = 0

else:

patResult = 1

if (patternValid[i, 0] == 1 and patternValid[i, 0] == patResult):

positivo = positivo + 1

elif (patternValid[i, 0] == 0 and patternValid[i, 0] == patResult):

negativo = negativo + 1

elif (patternValid[i, 0] == 1 and patternValid[i, 0] != patResult):

falsoNegativo = falsoNegativo + 1

elif (patternValid[i, 0] == 0 and patternValid[i, 0] != patResult):

falsoPositivo = falsoPositivo + 1

print("Positivo: %d" % positivo)

print("Negativo: %d" % negativo)

print("Falso Positivo: %d" % falsoPositivo)

print("Falso Negativo: %d" % falsoNegativo)

print("\n")

positivoTotal = positivo + falsoNegativo

negativoTotal = negativo + falsoPositivo

percentPositivo = positivo / positivoTotal \* 100

percentNegativo = negativo / negativoTotal \* 100

percentFalsoPositivo = falsoPositivo / negativoTotal \* 100

percentFalsoNegativo = falsoNegativo / positivoTotal \* 100

accuracy = ((positivo + negativo) / numPatValid) \* 100

recall = (positivo / positivoTotal) \* 100

precision = (positivo / (positivo + falsoPositivo)) \* 100

print("Porcentaje de aciertos positivos: %3.2f%%" % percentPositivo)

print("Porcentaje de falsos negativos: %3.2f%%" % percentFalsoNegativo)

print("Porcentaje de aciertos negativos: %3.2f%%" % percentNegativo)

print("Porcentaje de falsos positivos: %3.2f%%" % percentFalsoPositivo)

print("\n")

print("Accuracy: %3.2f%%" % accuracy)

print("Recall: %3.2f%%" % recall)

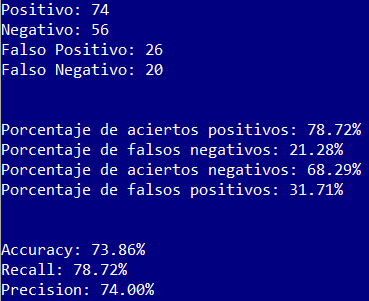
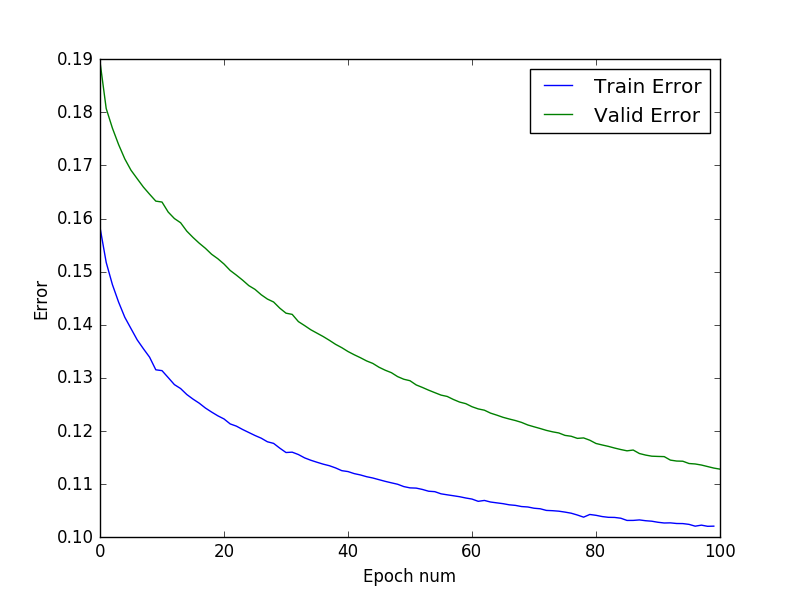
print("Precision: %3.2f%%" % precision)

Para las pruebas iniciales se usarán 25 neuronas ocultas, 0,0001 de Learning Rate y 0,1 de Momentum, llegando a un máximo de 100 iteraciones de entrenamiento. Estas pruebas servirán como trampolín a análisis posteriores, teniendo una referencia de partida que intentar mejorar con dichos análisis. Los porcentajes obtenidos serán calculados con el dataset de validación.

* 1. Cáncer de mama

El dataset de validación para el cáncer de mama contiene un 53,41% de casos positivos y un 46,59% de casos negativos.

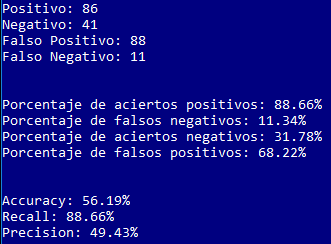
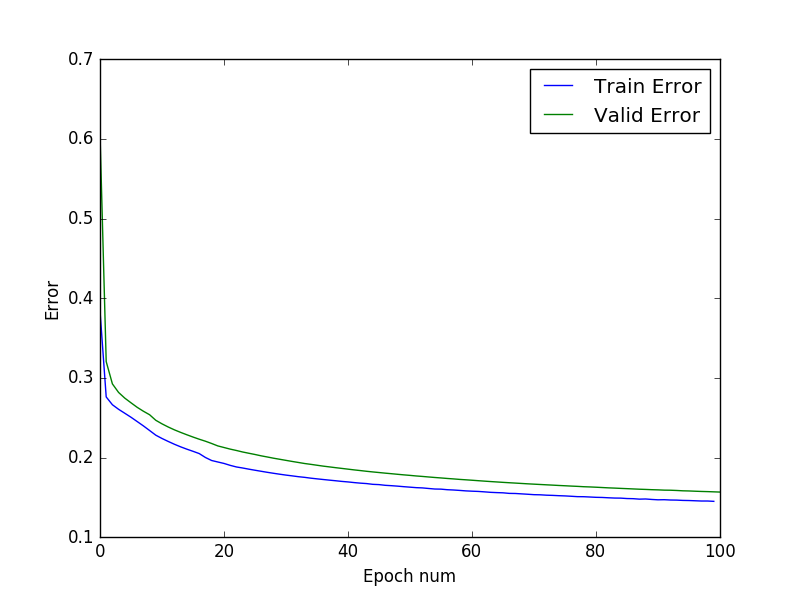
Los resultados fueron los siguientes:



* 1. Melanoma

La base de datos de validación para este tipo de cáncer contiene un 42,92% de pacientes con cáncer y un 57,08% de pacientes sin cáncer.

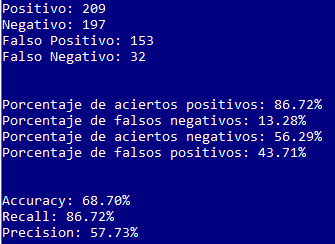
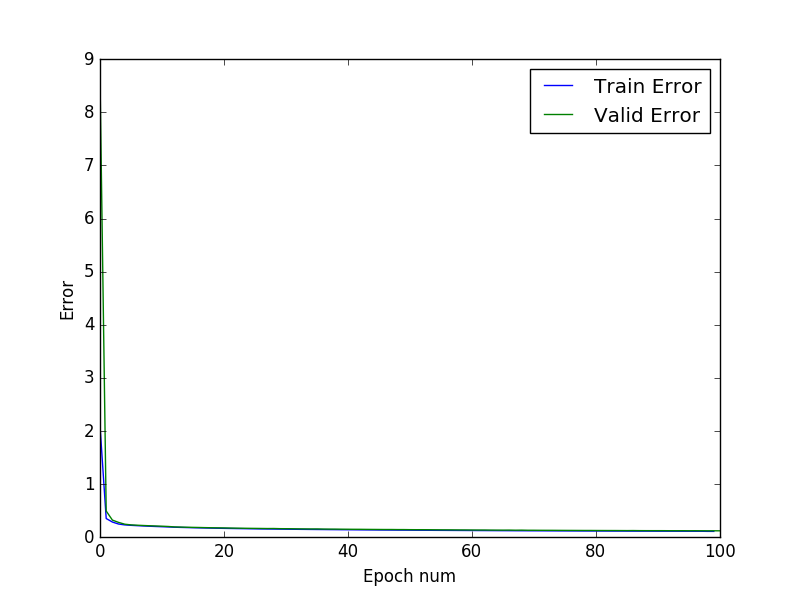
Se han obtenido los siguientes resultados:



* 1. Cáncer de pulmón

La base de datos para el cáncer de pulmón es la más extensa de las tres por lo que se esperan los mejores resultados de todos los casos. El conjunto de validación con el que se llevarán a cabo los análisis tiene un 40.78% de pacientes con cáncer y un 59.22% de pacientes sin cáncer.

Los resultados fueron los siguientes:



En general los resultados son satisfactorios para ser pruebas iniciales, si bien es cierto que en el caso del cáncer de pulmón aunque tenga porcentajes de accuracy menores a los del cáncer de mama se espera que durante el análisis la mejora sea mayor que la experimentada por este y obtenga mejores resultados, debido a la gran cantidad de datos disponibles.

Ahora el objetivo será mejorar estos porcentajes y mejorar la gráfica de aprendizaje intentando conseguir menores errores en menor tiempo modificando para ello las variables necesarias del algoritmo de aprendizaje y la red neuronal.

1. Análisis de neuronas ocultas

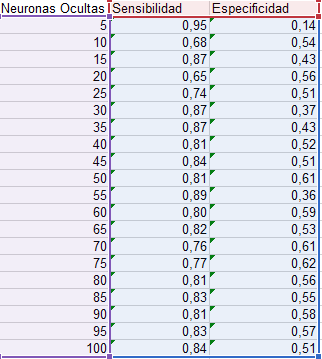
A continuación, se procederá a hacer pruebas con varios números distintos de neuronas ocultas, para observar en cuál de ellos la red neuronal tiene un mejor resultado. Las pruebas se realizarán modificando el número de neuronas ocultas de 5 a 100 con pasos de 5 neuronas (5, 10, 15,…,100), pero dejando el resto de variables con el mismo valor que en la prueba inicial, es decir, un Learning Rate de 0.0001 y un Momentum de 0.1, ya que estas variables influyen en el aprendizaje de la red y no en la red en sí misma.

Para cada número de neuronas se harán 5 pruebas obteniendo la media de ellos como representante de la instancia en cuestión.

Las pruebas se realizarán usando los valores de Sensibilidad y Especificidad. La sensibilidad indica la capacidad del estimador para dar como casos positivos los casos realmente enfermos, es decir, la proporción de enfermos correctamente identificados. Mientras que la especificidad indica todo lo contrario, la proporción de sanos correctamente identificados.

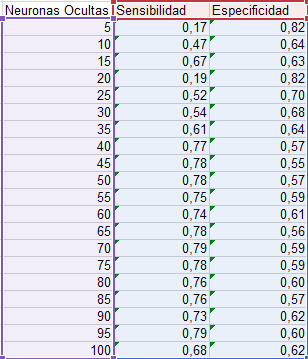
En este caso se buscará un valor de sensibilidad lo más alto posible sin despreciar tampoco el valor de especificidad, ya que es bastante más importante detectar correctamente a un paciente que tiene cáncer que detectar como sano a un paciente que no lo tiene.

* 1. Cáncer de mama



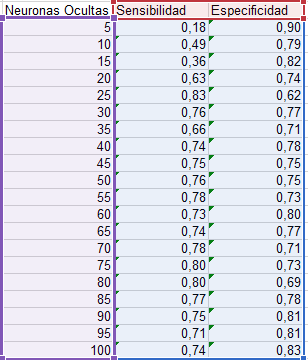
Se ha decidido **usar 50 neuronas ocultas**. A pesar de no ser la sensibilidad más alta de la tabla, tiene la mejor relación, entre las dos variables, ya que. sin perder mucha sensibilidad con respecto a los mayores valores, se gana bastante especificidad.

* 1. Melanoma



Los resultados en el caso del melanoma son peores que el cáncer de mama en cuanto a sensibilidad, ya que aquí no superan el 80%. Esto puede deberse a la mayor existencia de casos de pacientes sanos que pacientes con cáncer en el conjunto usado para entrenar la red. Se elegirá **usar 95 neuronas ocultas**, esta instancia obtuvo la mejor sensibilidad y una muy buena especificidad en comparación.

* 1. Cáncer de pulmón



Obviamente por tener más casos con los que entrenar la red neuronal, se demuestra que los resultados son mejores, no tanto en cuanto a sensibilidad, pero sí que se nota una gran diferencia en Especificidad. Se ha decidido usar **75 neuronas ocultas**¸ debido a que tiene la mayor sensibilidad, descartando el resultado con mejor relación entre ambas variables, que es el caso de las 100 neuronas ocultas, por considerar que dicha sensibilidad es muy baja, por lo que se prefirió perder 10 puntos de especificidad por ganar 6 en sensibilidad.

1. Análisis de Learning Rate

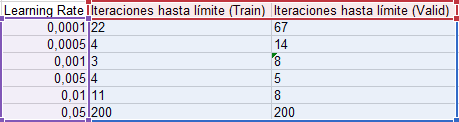
A continuación se realizará el análisis de la variable Learning Rate. Hasta ahora ha tenido un valor de 0.0001, pero se procederá a realizar pruebas para obtener el mejor resultado posible. Las pruebas consistirán en hallar con qué valor de Learning Rate se alcanza un error determinado en menos tiempo.

En nuestro caso, observando valores de error en las pruebas previas se ha decidido que dicha cota sea ‘0.13’, por lo cual el script calculará la iteración en la que el error baja de dicho valor. Existen casos en los que el error no llega a bajar de ese valor durante las iteraciones que tiene determinadas el entrenamiento, por lo cual se harán 5 pruebas por cada valor de Learning Rate usando el mejor de ellos como representante de esa instancia intentando ignorar los casos en los que no se llegue a ese límite en el caso del error de entrenamiento como en el error de validación.

Se ha decidido limitar el entrenamiento a 200 iteraciones, ya que aproximadamente a esa altura el valor de error será continuo. Se analizará el comportamiento de la red neuronal con los siguientes valores de Learning Rate:

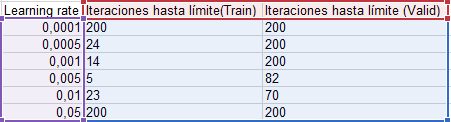
*0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05* (Se pretendía realizar pruebas con valores mayores, pero la red neuronal provocaba un overflow).

* 1. Cáncer de Mama



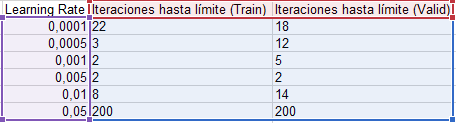
**Se utilizará 0.005 de Learning Rate.** Con este valor se obtuvo el menor número de iteraciones para el error de validación (5) y el segundo mejor para el error de entrenamiento (4).

* 1. Melanoma



En este caso se ven más dificultades para alcanzar la cota de error de 0,13 que con el cáncer de mama. Se ha decidido seleccionar **0,005 de Learning Rate**, al igual que en caso anterior, ya que, a pesar de no tener el mínimo de iteraciones para el error de validación, sí que tiene el mínimo de iteraciones para el error de entrenamiento y la relación es mejor.

* 1. Cáncer de pulmón



De nuevo para el este cáncer se elegirá usar 0.005 de Learning Rate, obteniendo los mejores resultados en general, tal y como se esperaba con 2 iteraciones para alcanzar la cota, tanto en error de entrenamiento como en error de validación.

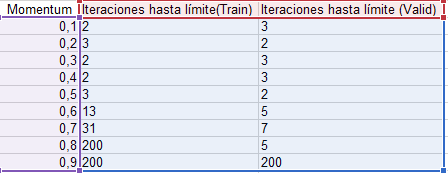
En conclusión, era de esperar que los tres tipos de cáncer tuvieran el mismo o parecido Learning Rate, ya que es un valor que suele depender del número de entradas y salidas de la red neuronal, por lo que al ser las mismas para los tres tipos, debían coincidir.

1. Análisis del Momentum

A continuación, se realizará el análisis de la única variable que falta para el algoritmo de Backpropagation. Dicho análisis será muy similar al realizado con el Learning Rate, ya que se fijará como objetivo en reducir el número de iteraciones posibles.

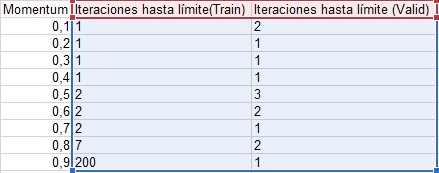
En esta ocasión se realizarán estudios desde 0.1 hasta 0.9 de Momentum, como siempre con 5 iteraciones para cada instancia, seleccionando el mejor como representante de la misma.

* 1. Cáncer de mama



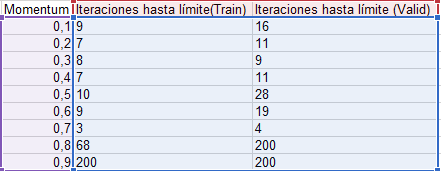
Durante los cinco primeros casos de estudio, los resultados son prácticamente idénticos, variando entre 3 y 2 iteraciones, tanto para entrenamiento como para validación, pero en el caso de 0,1 los resultados de las cinco pruebas son más homogéneos por lo que se ha decidido usar ese valor.

* 1. Melanoma



Como en el caso anterior las primeras pruebas son bastantes parecidas, pero elegiremos alguna de las tres que tardan solo una iteración para ambos errores (0.2, 0.3 y 0.4). Teniendo en cuenta el resto de pruebas realizadas para cada uno de esas tres instancias, la más homogénea es 0.3.

* 1. Cáncer de Pulmón



En este último caso del cáncer de pulmón, los resultados son más dispares que los anteriores y solo hay un resultado mejor que todos los demás, para la instancia de **0,7 de Momentum**.

En conclusión, comparando todos los casos entre sí, se puede ver menos igualdad que en el caso del estudio del Learning Rate, donde el resultado para los tres fue el mismo. Se puede decir que el Momentum será mayor cuando mayor sea la cantidad de datos a entrenar por la red.

1. Conclusiones