Universidad de Chile
Facultad de Cs. Físicas y Matemáticas
Departameto de Ciencias de la Computación
CC5114 - Redes Neuronales y Programación Genética

Tarea 1

Red Neuronal

Autor: Matías Meneses C.
Profesor: Alex Bergel
6 de septiembre de 2017
Santiago, Chile.

Índice

1. Introducción

El objetivo de esta tarea es programar una red neuronal que permita la clasificación o predicción de clases luego de un entrenamiento en la red neuronal, con un dataset dado.

Luego de entrenar la red, se realizarán pruebas para comprobar la efectividad del entrenamiento. Se realizará una variación en los parámetros de la red para analizar su impacto.

1.1. Software utilizado

Se utilizó el lenguaje de programación Python para realizar la implementación de la red neuronal, en conjunto con la librería unittest para la realización de test unitarios, y matplotlib para los gráficos.

El programa resultante fue ejecutado en una máquina con Linux (Fedora 25), intel core i5-4200U con 8 GB de RAM.

2. Preparación de Input

Se cuenta con un set de datos de 100 filas y 9 características, que corresponde a un test de fertilidad en muestras de semen (`https://archive.ics.uci.edu/ml/datasets/Fertility`). El output del test es N, si el diagnóstico es normal, u O, si el diagnóstico está alterado. Se procesó el set de datos de forma que el output fuera representado por un par $[x, y]$, donde el output será $[1, 0]$ si el diagnóstico es normal, y $[0, 1]$ si está alterado.

Se divide el set de datos en dos subconjuntos: El set de entrenamiento, que corresponde al 70 % del set de datos, y el set de pruebas, que corresponde al 30 % restante. Estos subconjuntos son extraídos realizando una permutación aleatoria en el set de datos.

```
import random as rd

data_set = parse_dataset("dataset/fertility.csv")
rd.shuffle(data_set)

training_set = data_set[:int(len(data_set)*0.7)]
test_set = data_set[int(len(data_set)*0.3):]
```

3. Entrenamiento

Se utiliza el algoritmo de backpropagation para entrenar la red neuronal, con el sigmoid como función de activación.

4. Clasificación

Se clasifica el conjunto de pruebas analizando el output de las neuronas del último layer de la red. La clase que predice la red corresponderá a la neurona con el mayor valor entre las neuronas del layer.

5. Experimentos

El experimento se realizó variando la cantidad de entrenamiento realizado, desde un solo dato de entrenamiento, hasta el set de entrenamiento completo. Es importante la permutación aleatoria del set de entrenamiento, para evitar posibles problemas de representatividad al escoger un subconjunto. Se repitió el experimento para valores entre $0{,}1$ y $1{,}0$ del learning rate de la red neuronal, para comprobar su efecto. Por último, se realizaron dos construcciones distintas de la red neuronal, una con sólo un hidden layer con una neurona, y otra con dos hidden layers y dos neuronas en cada una. El input layer siempre tiene 9 neuronas, y el output siempre tiene dos, de acuerdo al número de características y outputs respectivamente.

Se calcula el porcentaje de aciertos de la red, repitiendo el experimento 100 veces para cada sample y calculando el valor promedio.

El análisis realizado se puede reproducir ejecutando el archivo *Predictor.py*. Dentro de este archivo se puede configurar la red neuronal utilizada y variar los experimentos.

6. Resultados

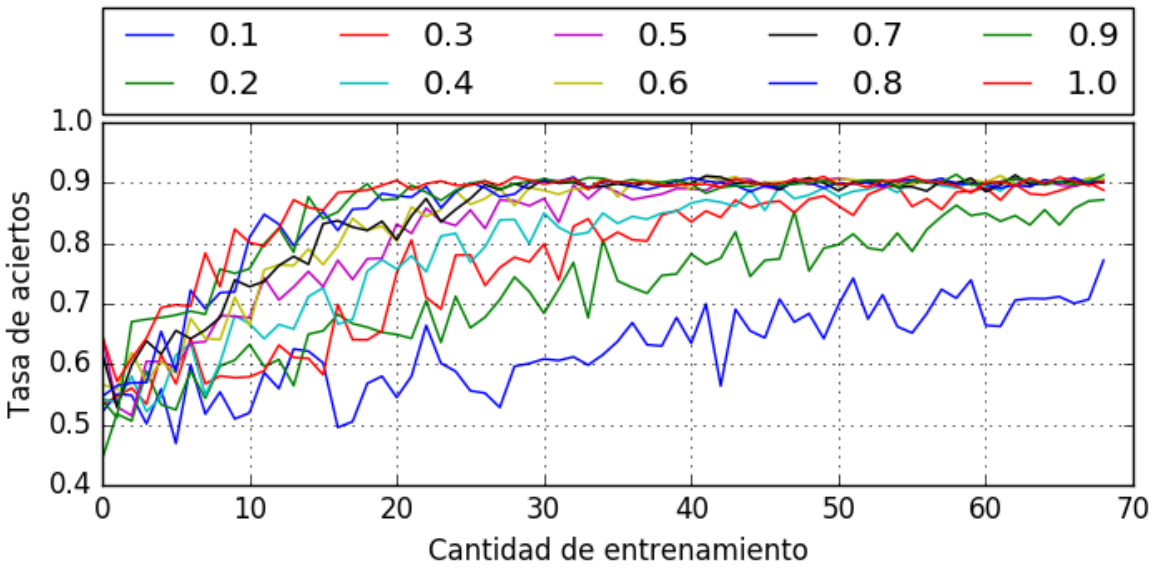

Figura 1: Experimento con una hidden layer para distintos valores de learning rate

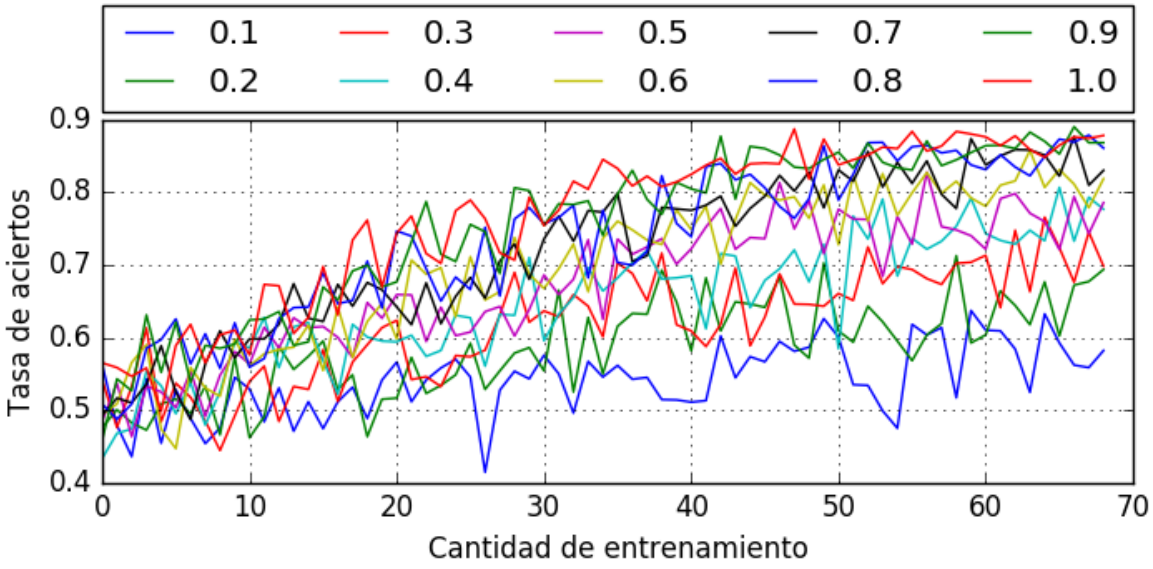

Figura 2: Experimento con dos hidden layer para distintos valores de learning rate

7. Análisis de Resultados

Se observa de los resultados obtenidos que la red neuronal logró predecir de mejor forma con más entrenamiento. El valor de 0.5 para pocos datos de entrenamiento se esperaba, pues en esta fase la red neuronal se comporta de forma aleatoria. Se observa una tasa de acierto que asciende hasta $0{,}9$ con el número de entrenamientos adecuedo.

Se observa que una variación en la estructura de la red neuronal impacta en la velocidad con que aprende la red. La red con dos hidden layers aprendió más lento que la red con sólo un hidden layer. Esto se justifica en que la propagación del error es más directa en el caso de menos layers, pero puede ser que esta construcción tenga mayor sesgo y no funcione correctamente ante la presentación de una mayor cantidad de casos de prueba.

Con respecto a la variación del learning rate, la velocidad de aprendizaje aumenta con este. En el caso de la red neuronal con un hidden layer, se observa que la diferencia entre learning rates mayores a 0.5 comienza a ser despreciable.

El experimento completo tomó 6 minutos en ejecutarse. Considerando que este dataset era pequeño, se concluye que la red no es óptima en tiempo de ejecución, y que se podría optimizar utilizando operaciones de matrices en vez de un enfoque orientado a objetos y recursividad.

8. Conclusión

Se concluye que la red neuronal fue capaz de aprender sobre el problema de las muestras de semen e infertilidad, logrando predecir hasta el 90 % de los casos del set de pruebas.

Se concluye que el *learning rate* y la cantidad de *hidden layers* impacta en la tasa de aprendizaje de la red.