Informe del Trabajo Práctico 3: Machine Learning

Agüero, Emanuel Bustillo, Carlos Lezcano, Agustín

04 de septiembre de 2020

1. Introducción

El objetivo del trabajo práctico es implementar una red neuronal feedforward fully connected de 1 capa oculta para poder clasificar tres grupos distintos generados de manera ficticia, mediante la implementación de un MLP.

El número de entradas son dos (según los ejes de las coordenadas cartesianas) y al ser tres grupos, las salidas son 3.

2. Accuracy y Loss

Se busca el máximo de cada fila en la salida real y se obtiene un vector de 300x1 (se encuentra el índice donde está el valor máximo de la lista). Posteriormente se compara fila por fila el valor del vector (que es el máximo de cada fila de) con la salida target donde coincida. Si coincide le asigna el valor 1, sino 0. Finalmente se realiza el promedio de las clases acertadas sobre el total de ejemplos tomados para obtener la eficiencia.

El valor de la función de Loss se usa para variar los pesos, cuyos valores cambian según el ritmo de aprendizaje y la función de Loss propiamente dicha. La misma se calcula como el promedio de los logaritmos de los valores P, de la siguiente forma:

$$L = \frac{1}{m} * \sum(-log(\frac{e^(y^m)_c}{\sum_j (e^(y)_j)} \tag{1}$$

3. Lerning rate, loss y parada temprana

Se utilizó una pequeña fracción de datos para la validación y para el test. Para la validación se evaluó el desempeño cada n *epochs*, comparando el valor de Loss entre el *dataset* de validación y el de training.

El *learning rate* o tasa de aprendizaje es un parámetro de ajuste en un algoritmo de optimización que determina el tamaño del paso en cada iteración mientras se mueve hacia un mínimo de una función de pérdida. En otras palabras, representa la magnitud con el que se mueve en el gradiente mientras que el loss define la dirección.

Se aplicó K fold para tener los parámetros ya optimizados al momento de evaluar el error entre el loss obtenido del Train y el loss obtenido en Validation. Y en caso de que el error absoluto supere el valor de tolerancia deja de estar evaluando.

Cabe aclarar que cuando se evaluó el K fold, en una primera etapa se varió el learning rate manteniendo fijos los epochs. Luego definido el learning rate se deja fijo y se varían los epochs.

Aclaramos que si se empieza a calcular el error absoluto a partir de un epochs = 0, el error que se comete supera el valor de la tolerancia, por eso solo marca un punto. Por esa razón propusimos que empiece a evaluar el error absoluto entre ambas curvas de lossValidation y lossTrain a partir del 10 % de lo epochs evaluados, para poder dejar que las curvas se estabilicen .

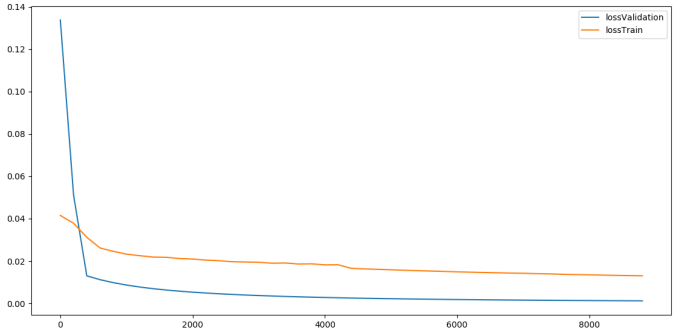


Figura 1: Comparar loss del Train vrs Loss de Validation para un valor de tolerancia = 0.3

4. Test: comparación para distintas funciones de datasets

La idea del test es probar el funcionamiento del algoritmo con los valores que no aprendió antes. Se realizó con el valor de los pesos ya dados mediante el training, o sea que para realizar el test se modifican los mismos. Para el test se creo otro dataset distinto y realizó el *feed forward* una vez para los ejemplos dados, esto quiere decir que se evaluó la función en un *epoch*. Para la evaluación del test se usan métricas de *accuracy*, en este caso se usó el ratio (cantidad de aciertos / cantidad de tests) (esto representa el porcentaje de aciertos).

Para distintas evaluaciones (test) de los algoritmos usando la función ReLU los resultados de *accuracy* fueron del orden superior a 90 %. En cambio, usando una función sigmoidal los resultados de *accuracy* son superiores a 75 %, lo cual indica que según la función sigmoidal planteada y comparando con la función

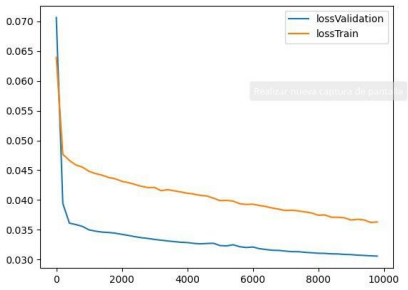
ReLU, la segunda es más eficiente que la primera. Este problema se puede deber al *gradient vanishing*, que significa que debido al gradiente de la función pequeño hacen falta muchas iteraciones para poder lograr valores aceptables de eficiencia durante el entrenamiento.

Para poder visualizar la comparación entre los resultados obtenidos usando una u otra función para crear el dataset se realizó un cuadro comparativo (ver tabla 1). Se debe notar que los valores pueden diferir al ejecutar nuevamente el programa ya que los valores generados cada vez que se ejecuta el programa son aleatorios.

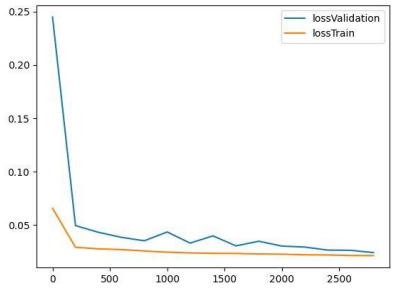
Eficiencia en training(reLU)	Eficiencia en test(reLU)	Eficiencia en training(sigmoide)	Eficiencia en test(sigmoide)
0.9933333333333333	0.9333333333333333	0.7533333333333333	0.7333333333333333
1.0	1.0	0.7533333333333333	0.75
1.0	0.9666666666666667	0.8366666666666667	0.7333333333333333
0.9966666666666667	0.85	0.77	0.7333333333333333
0.99	0.9	0.77	0.7666666666666667
1.0	0.9666666666666667	0.77	0.85

Cuadro 1: Comparación de los resultados obtenidos para distintas funciones

Al realizar el *tunning* de parámetros se pudo observar que la respuesta obtenida mejora notablemente con la adición de neuronas en la capa oculta, pudiendo evitar el overfitting al aumentarlas en número.



(a) 100 neuronas en la capa oculta



(b) 140 neuronas en la capa oculta

5. Nuevo generador de datasets

Se propuso la alternativa de funciones gaussianas, para tener un mayor solapamiento entre las clases. Si se desea tener un mayor solapamiento solo se debería modificar la desvición estándar.

Para el Set de datos original se obtuvieron los siguientes resultados 3):
Eficiencia en training 0.9933333333333333
Eficiencia en test: 0.9333333333333333

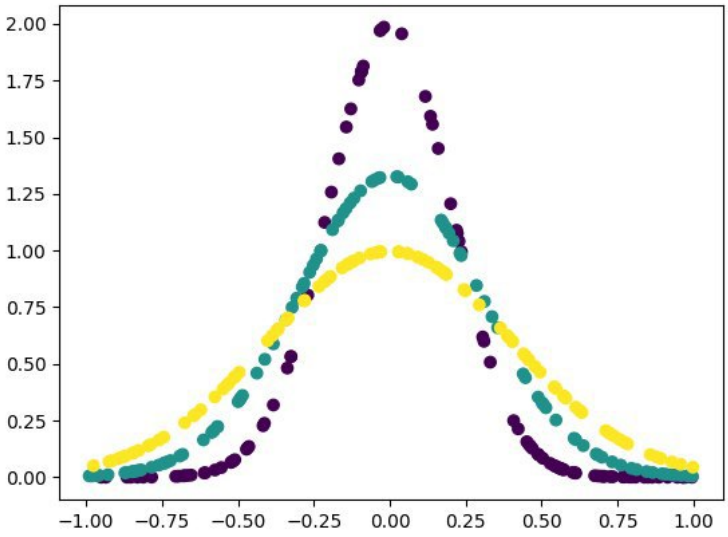
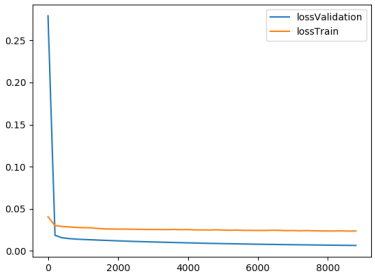
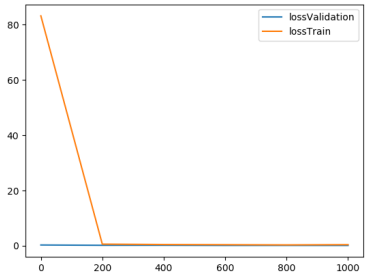


Figura 3: Nuevo conjunto obtenido en el plano

Para el nuevo set de datos se obtuvieron los siguientes resultados (**??**):
Eficiencia en training 0.9466666666666667
Eficiencia en test: 0.8



(a)
Para set de datos original



(b)
Para el nuevo set de datos

6. Regresión

Se representaron un dataset de funciones continuas que posteriormente se discretizaron ciertos puntos para analizar. Se consideró una sola entrada.

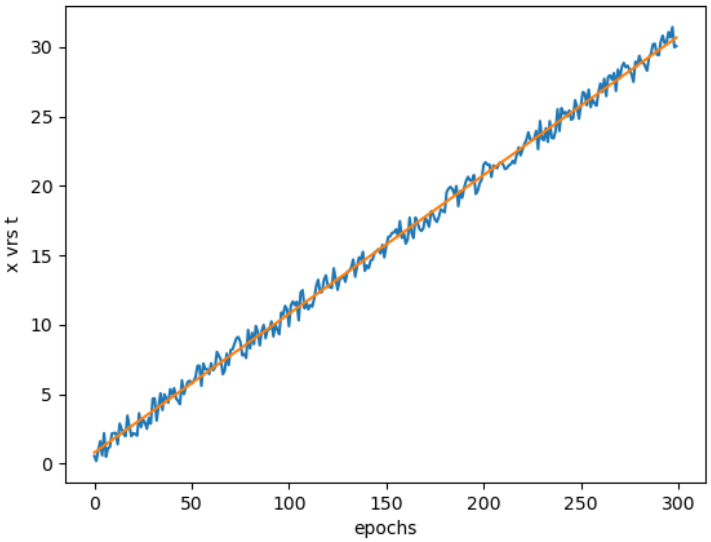


Figura 5: Representación para el dataset de funciones continuas

Nota: Debido a las dimensiones de las matrices utilizadas y utilizar tanto en las funciones como en sus respectivas derivadas, al momento de utilizar la transpuesta tal como se plantean en los apuntes de la cátedra, se generan conflicttos debido a que en cada iteración hace que las matrices no cuadradas cambien filas por columnas y viceversa, produciendo fallas al momento de representar la resolución del ejercicio. Detectamos dónde se encuentra el error, pero no logramos encontrar una solución a ese pequeño detalle. Consideramos que esto es una deuda técnica con nosotros mismos, y que en un futuro trataremos de solucionar para lograr implementarlo en nuestra vida profesional.