Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Técnicas de inteligencia artificial

Reporte: Práctica I del perceptrón multicapa

Docente: Abraham Sánchez López

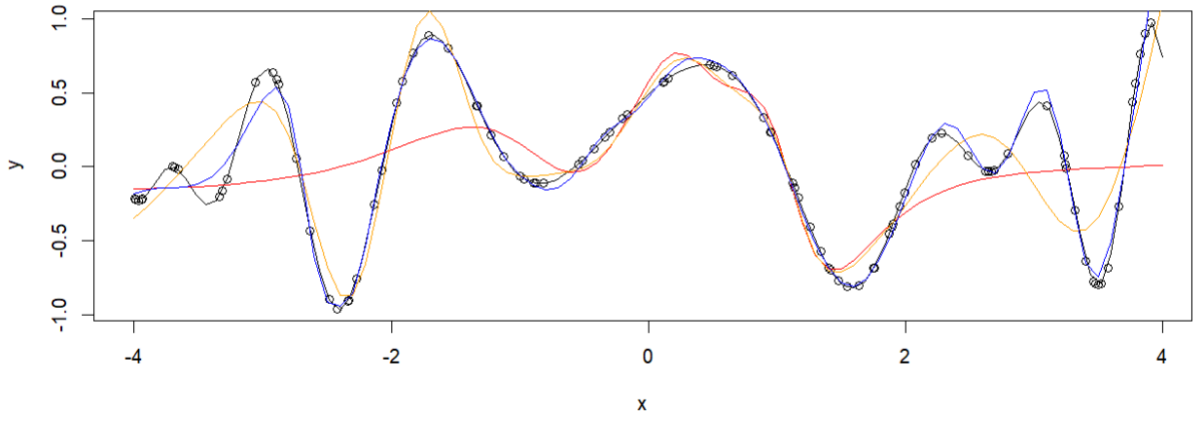
Alumno	Matrícula
Taisen Romero Bañuelos	202055209

Práctica I del perceptrón multicapa

A manera de repaso, dada una función matemática f(x) se generan de manera aleatoria valores de f(x) para que estos sean dados a un modelo como datos al azar que tiene que interpretar para descubrir la forma de la función f(x) que desconoce. Bien, en principio una red neuronal capaz de replicar fielmente a f(x) en un plano cartesiano debería ser suficiente para que digamos que tenemos un buen modelo, pero me surgió la duda de cómo reaccionaría este modelo supuestamente óptimo si introducimos datos nuevos. Como tal, dado que grafica muy bien a f(x) en el intervalo de las abscisas que va de [-4,4] de poco serviría darle un valor nuevo aleatorio que se encuentre dentro de este rango numérico, por lo cual, yo propongo que se expanda dicho rango de [-4,6]. Nuestra función original seguirá su curso normal y nuestro modelo deberá revelarnos si fue capaz de generalizar adecuadamente las entradas o simplemente se ajustó lo suficiente para el rango anterior.

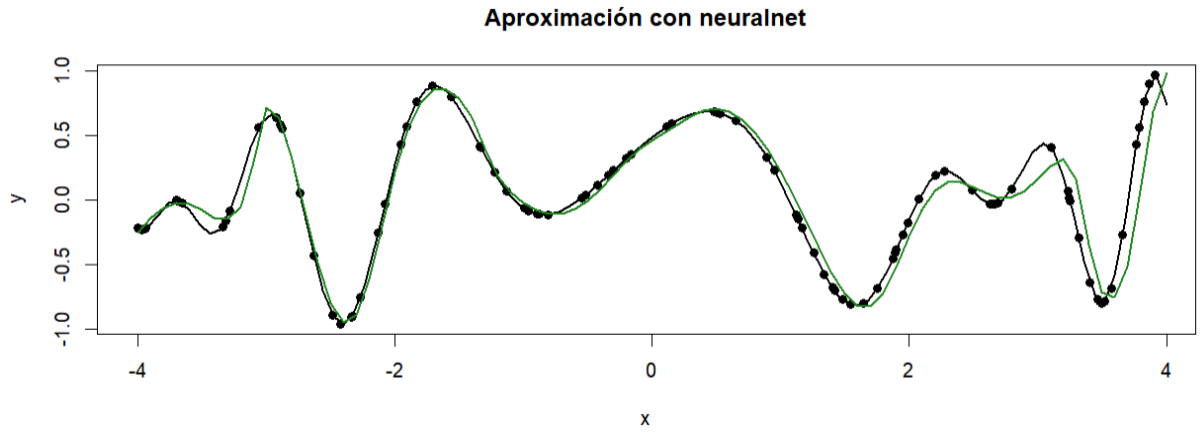
Antes de pasar a esa evaluación haré un repaso de los resultados que obtuve con las combinatorias.

Con nnet:



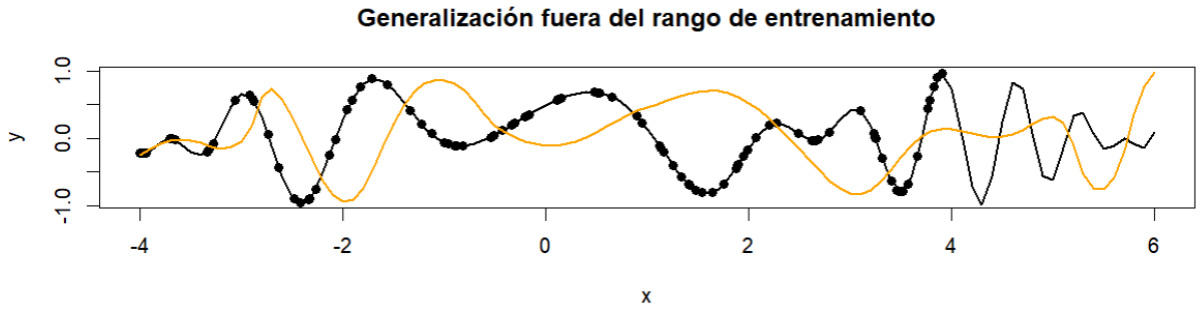
La línea negra es f(x); la línea roja es la configuración del PDF; la línea naranja es una experimentación inicial; y la línea azul es la configuración que mejores resultados me dio (60 neuronas y 200 iteraciones).

Con neuralnet:

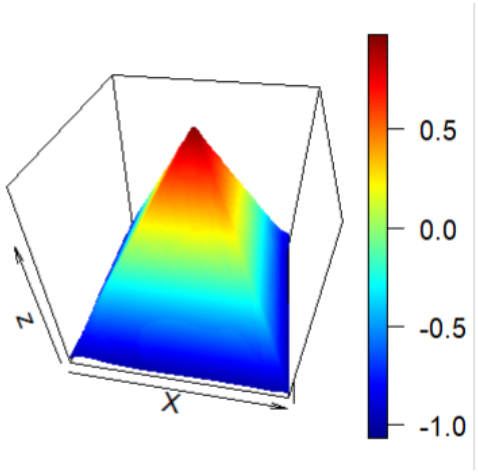


Con neuralnet hice varias combinaciones entre el número de capas y neuronas. Noté que el tiempo de ejecución aumentaba notablemente a partir de las 70 neuronas en tan sólo dos capas, sin embargo, el tiempo de ejecución era más eficiente si esas 140 neuronas de las dos capas las dividía en más capas. Es decir, aumentar el número de capas es menos costoso siempre y cuando el número de neuronas por capa no sea tan grande. Mis mejores combinaciones fueron las siguientes (daban casi el mismo resultado, sólo variaron en tiempo): El mejor - hidden=**c(20,20,20,20)**; Tarda un poco - hidden=c(70,70); hidden=c(40,40,40); hidden=c(35,35,35,35).

Bien, para la expansión del rango numérico usé el modelo generado con el paquete de neuralnet ya que me gustó más su exploración de valores. Como se verá a simple vista, pese a que usé el mismo modelo con la misma configuración las predicciones de -4 a 4 se vieron alteradas, y no sólo eso, también pareciera que no generalizó porque las predicciones de 4 a 6 no coinciden en absoluto. Pero si nos detenemos a observar con detenimiento podemos notar que la función naranja (las predicciones) son la misma función predicha de -4 a 4 con la diferencia de que está estirada sobre el eje de las x. Esto se puede interpretar como que la red está aplicando la misma estructura de predicción que aprendió en el rango de entrenamiento (de -4 a 4), pero al extrapolar a valores que nunca vio (de 4 a 6), la red repite patrones aprendidos sin ajustarlos al nuevo contexto. Este comportamiento puede explicarse por el hecho de que la red no ha recibido ejemplos en ese rango, y por lo tanto, no tiene manera de saber cómo debería comportarse la función real ahí. O bien podemos decir que simplemente está sobre ajustado, pero me gusta más verlo de la forma anterior.



Y sobre el inciso B esta fue mi mejor configuración:



Mi número de iteraciones fue algo excesivo, pero se debe a que noté que obtenía mejores resultados si aumentaba más las iteraciones que el número de neuronas. Usé 100 neuronas y 5,000 iteraciones.