Diseño Evolutivo de Redes Neuronales con Evolución Diferencial

Alfredo Gutiérrez Alfaro

Introducción

- A la hora de implementar redes neuronales para resolver un problema requiere de personas expertas para diseñar la topología de la red y paramétros, entre otros elementos de diseño. (López-Vázquez et al. 2019)
- Para resolver esta problemática del diseño, en la literatura se puede encontrar el uso de metaheuristicas bioinspiradas, como lo es el uso del Particle Swarm Optimization (PSO), Ant-Colony y la Evolución Diferencial. (Garro and Vázquez 2015; López-Vázquez et al. 2019; Alba-Cisneros et al. 2020)

Objetivo

- Implementar y analizar la evolución diferencial para el diseño de redes neuronales y cómo se desempeñan en las tareas de clasificación.
- Comparar resultados con una red neuronal “tradicional” entrenada con el descenso del gradiente

Redes Neuronales

Las redes neuronales son un modelo matemático que intentan replicar de cierta forma a las neuronas biológicas.

$$\sigma(\sum_{i=1}^{m} x_i w_i + b) = \sigma(x^T w + b) = \hat{y} \quad (1)$$

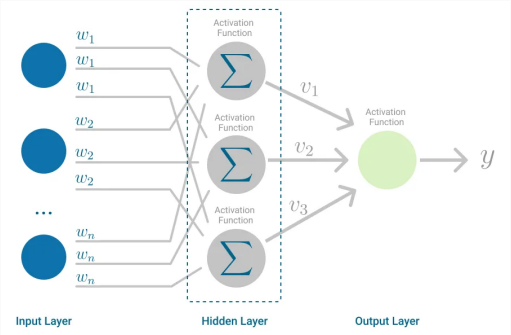

Figura 1: Ejemplo red neuronal (Bento 2022)

Funciones de activación

Las funciones de activación introducen no linealidad dentro de una red neuronal, lo que le permite a la red realizar representaciones más complejas de los datos.

Nombre	Función
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Sinusoidal	$\sin(x)$
Linear	$f(x) = x$
Hard Limit	$f(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$
ReLU	$f(x) = \text{máx}(0, x)$
Leaky ReLU	$f(x) = \begin{cases} 0{,}1x & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$

Aprendizaje evolutivo

Los algoritmos evolutivos son un tipo de algoritmos de optimización heurísticos y aleatorizados, inspirados en la evolución natural. Simulan el proceso de evolución natural considerando dos factores clave: la reproducción variacional y la selección del más apto. (Zhou, Yu, and Qian 2019)

Los estructura básica de la mayoría de algoritmos evolutivos se puede resumir de la siguiente manera:

1. Generar un conjunto inicial de soluciones (llamado población).
2. Reproducir nuevas soluciones basadas en la población actual, mediante procesos como el cruce y la mutación.
3. Eliminar las peores soluciones de la población.
4. Repetir desde el paso 2 hasta que se cumpla algún criterio de parada.

Evolución diferencial

La evoluación diferencial es un algoritmo evolutivo que sirve para resolver problemas continuos. La ED utiliza una estrategia multiparental para generar posibles soluciones. El algoritmo se basa en la reproducción de uno o más individuos, reemplazando a los padres por hijos con mejor aptitud. (Du and Swamy 2016)

Pseudocódigo

Algorithm 1: Differential Evolution (DE)

Input : Number of individuals NP

Output: Optimized solution P

```
1  Generate P = (x1, x2, ..., xNP);
2  repeat
3      for i = 1 to NP do
4          Compute a mutant vector vi;
5          Create ui by the crossover of vi and xi;
6          if f(ui) < f(xi) then
7              Insert ui into Q;
8          end
9          else
10             Insert xi into Q;
```

Representación del problema

Metodología

Diseñar una red neuronal de tres capas con algorimos evolutivos (evolución diferencial) con un esquema de codificación directo (Garro and Vázquez 2015), que tiene en consideración los siguientes elementos:

1. Definir el número de neuronas en la capa oculta
2. Establecer funciones de activación
3. Generar conexiones sinapticas y pesos

Definir el número de neuronas en la capa oculta

$$Q = (M + N) + \frac{N + M}{2} \tag{2}$$

$$H = Q - (M + N) \tag{3}$$

$$dim_d = [H * (N + 3)] + [M * (H + 3)] \tag{4}$$

Representación del problema

A continuación una muestra de cómo se representa la codificación para el diseño de las redes neuronales:

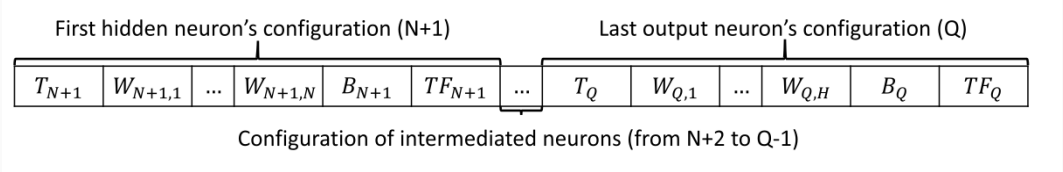

Figura 2: Esquema para representar los paramétros (Alba-Cisneros et al. 2020)

Métricas

Para poder evaluar la red y utilizar una función a optimizar dentro de la evolución diferenciar se hará uso de la exactitud y el error de esta:

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

$$error = 1 - exactitud \tag{6}$$

Experimentos

Se utilizaron las ecuaciones [2, 3, 4] para generar las neuronas (arquitectura) de una red neuronal para dos conjuntos de datos diferentes: Iris Plant (Fisher 1988) y Wine (Aeberhard and Forina 1991).

Posteriormente esta red fue codificadas para ser utilizadas por la evolución diferencial y así obtener la topología de la red así como sus paramétros para después evaluar la red con la exactitud.

Resultados

Aqui pongo gráfica de las redes para ambos datasets

Resultados

Aquí agrego dos gráficas para comparar la exactitud de la evolución con scikit

Conclusión

Aqui pongo algo

Referencias

Aeberhard, Stefan, and M. Forina. 1991. “Wine.” UCI Machine Learning Repository.

Alba-Cisneros, O., A. Espinal, G. López-Vázquez, M. A. Sotelo-Figueroa, O. J. Purata-Sifuentes, V. Calzada-Ledesma, R. A. Vázquez, and H. Rostro-González. 2020. “Direct and Indirect Evolutionary Designs of Artificial Neural Networks.” In *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications*, edited by Oscar Castillo, Patricia Melin, and Janusz Kacprzyk, 431–43. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-35445-9_31.

Bento, Carolina. 2022. “Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis,” January. https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141.

Du, Ke-Lin, and M. N. S. Swamy. 2016. *Search and Optimization by Metaheuristics*. Birkhäuser.

Fisher, R. A. 1988. “Iris.” UCI Machine Learning Repository.