INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
INTELIGENCIA ARTIFICIAL
ING. EN SISTEMAS COMPUTACIONALES

Redes Neuronales de Base Radial

Alumno: Hurtado Morales Emiliano - 2021630390

Maestro: Andrés García Floriano

Grupo: 6CV3

Fecha de entrega: 16/12/2024

Ciclo Escolar: 2025 – 1

Contenido

1. Introducción

Las Redes Neuronales de Base Radial (RBF Networks)

Las Redes Neuronales de Base Radial (RBF Networks) son un tipo específico de red neuronal artificial que destaca por su capacidad de manejar problemas complejos de clasificación, regresión y aproximación de funciones. Estas redes son particularmente útiles en situaciones donde las relaciones entre los datos no pueden modelarse mediante separabilidad lineal en su espacio original, como ocurre en problemas de clasificación no lineal.

Una RBF Network se inspira en el funcionamiento de las redes neuronales del cerebro humano al emular procesos de aprendizaje y adaptación. A diferencia de las redes perceptrón multicapa (MLP), que dependen de funciones de activación no lineales como la sigmoide o ReLU, las RBF se caracterizan por utilizar **funciones de base radial**. Estas funciones se activan en función de la distancia radial entre una entrada y un punto central denominado centroide.

Transformación al Espacio de Alta Dimensión

La fortaleza de las RBF radica en su capacidad para transformar las entradas originales en un **espacio de mayor dimensión** mediante la aplicación de funciones de base radial. Este proceso permite que datos inicialmente no separables linealmente en su espacio original puedan dividirse de forma clara en el espacio transformado.

La transformación funciona de la siguiente manera:

1. **Entrada al espacio de alta dimensión**:
    - Las características de los datos originales se mapean a un espacio donde las funciones de base radial calculan la similitud de los puntos con respecto a un conjunto de puntos predefinidos (centroides).
2. **Separabilidad lineal**:
    - Una vez en el espacio transformado, las relaciones no lineales se convierten en separaciones lineales que pueden resolverse mediante métodos simples como la combinación lineal ponderada.

2. Arquitectura de una RBF Network

2.1 Capa de entrada

La capa de entrada actúa como un receptor que transfiere los datos sin transformarlos. Cada nodo de esta capa corresponde a una característica o atributo del conjunto de datos de entrada. Por ejemplo, si los datos representan flores con características como "longitud del pétalo" y "ancho del sépalo", cada nodo de la capa de entrada se corresponde con una de estas dimensiones.

2.2 Capa oculta

La capa oculta es el núcleo de las RBF y contiene neuronas que transforman los datos utilizando funciones de base radial. Estas neuronas miden la similitud entre el vector de entrada y ciertos puntos específicos en el espacio, llamados **receptores** o **centroides**.

Funciones de base radial (RBF)

La función de base radial más común es la **función Gaussiana**, definida como:

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Donde:

- $r = \|x - t\|$: es la distancia euclidiana entre el vector de entrada x y el centroide t.
- σ: es un parámetro que controla la dispersión de la función Gaussiana.

Otras funciones de base radial incluyen:

1. **Función multicuadrática inversa**:

$$\phi(r) = \frac{1}{\sqrt{r^2 + \sigma^2}}$$

2. **Función cúbica**:

$$\phi(r) = r^3$$

Cada neurona en esta capa genera una salida que indica qué tan cerca está el patrón de entrada del centroide asociado.

2.3 Capa de salida

La capa de salida combina las respuestas de la capa oculta mediante una **combinación lineal ponderada**. Su objetivo es generar la predicción final, que puede ser un valor continuo (regresión) o una clase (clasificación).

La salida de la capa de salida se calcula como:

$$z_i = \sum_j w_{ij}\phi(r_j) + \theta_i$$

Donde:

- zi: es la salida de la neurona i en la capa de salida.
- w_{ij}: es el peso que conecta la neurona j de la capa oculta con la neurona i de la capa de salida.
- $\phi(r_j)$: es la salida de la función de base radial de la neurona j.
- θ_i: es un término de sesgo ajustable.

3. Funcionamiento

3.1 Transformación en la capa oculta

1. **Cálculo de la distancia euclidiana**: Cada neurona oculta mide la distancia euclidiana entre el patrón de entrada x y su centroide t:

$$r_j = \|x - t_j\|$$

2. **Aplicación de la función de base radial**: Con la distancia r_j, se aplica la función de activación:

$$\phi(r_j) = \exp\left(-\frac{r_j^2}{2\sigma^2}\right)$$

Esto genera una salida que es mayor cuando el patrón está cerca del centroide.

3.2 Combinación lineal en la capa de salida

La capa de salida realiza una combinación lineal ponderada de las salidas de la capa oculta, utilizando los pesos w_{ij}. Esto permite obtener una salida que se utiliza para clasificar o aproximar valores.

4. Entrenamiento

4.1 Entrenamiento de la capa oculta

El objetivo principal es definir los **centroides** y los parámetros σ de las funciones de base radial.

Determinación de los centroides

1. Utilizar un algoritmo de clustering, como **K-Means**, para dividir los datos en M clusters.
2. Cada centroide t_j se define como el centro geométrico de un cluster.

Cálculo de σ

El parámetro σ se calcula como la media de las distancias cuadradas entre los puntos del cluster y su centroide:

$$\sigma_j^2 = \frac{1}{N_j} \sum_{x \in \text{cluster j}} \|x - t_j\|^2$$

Donde:

- N_j: número de puntos en el cluster j.

4.2 Entrenamiento de la capa de salida

El entrenamiento de esta capa ajusta los pesos w_{ij} para minimizar el error entre la salida deseada y la obtenida. Esto se realiza utilizando métodos como:

- **Regresión lineal**: para problemas de regresión.
- **Retropropagación**: para problemas de clasificación.

5. Ventajas y Desventajas

Ventajas

1. **Entrenamiento rápido**: Las RBF requieren menos iteraciones de entrenamiento en comparación con las MLP.
2. **Interpretabilidad**: Es más fácil interpretar el significado de los nodos ocultos porque representan regiones específicas del espacio de características.
3. **Resolución de problemas no lineales**: Las RBF son efectivas para problemas complejos al transformar los datos a un espacio de mayor dimensión.

Desventajas

1. **Sensibilidad a los parámetros**: La elección de los centroides y σ\sigmaσ afecta significativamente el rendimiento.
2. **Escalabilidad limitada**: Con conjuntos de datos grandes o de alta dimensionalidad, el entrenamiento y la clasificación pueden ser lentos.
3. **Necesidad de clustering eficiente**: El desempeño depende de la calidad del algoritmo de clustering utilizado para definir los centroides.

6. Aplicaciones

6.1 Clasificación

- Reconocimiento de patrones en imágenes.
- Clasificación de datos biométricos, como reconocimiento de huellas dactilares.

6.2 Aproximación de funciones

- Predicción de series temporales.
- Modelado de funciones complejas en sistemas físicos y financieros.

6.3 Sistemas de control

- Control adaptativo en robótica.
- Automatización industrial.

7. Ejemplo

En este caso, se trabajará con un problema de clasificación binaria en dos dimensiones usando datos generados artificialmente con dos clases bien definidas.

El problema utiliza el conjunto de datos de **círculos concéntricos** generado por make_circles de la biblioteca sklearn. Este es un ejemplo clásico de un problema **no linealmente separable**, adecuado para probar la capacidad de una red RBF para transformar datos al espacio donde son separables linealmente.

Explicación del Código

```
# Generar datos artificiales (dos círculos concéntricos)
def generar_datos_circulos():
    X, y = make_circles(n_samples=300, factor=0.5, noise=0.05, random_state=42)
    return X, y
```

1. Generación de Datos

- Se generan 300 muestras distribuidas en dos círculos concéntricos con ruido añadido. La variable factor controla el tamaño relativo de los círculos.

```
    def entrenar(self, X, y):
        # Paso 1: Encontrar centroides con K-Means
        kmeans = KMeans(n_clusters=self.num_centroides, random_state=42).fit(X)
        self.centroides = kmeans.cluster_centers_

        # Paso 2: Calcular sigma (media de las distancias entre los centroides)
        distancias = cdist(self.centroides, self.centroides, 'sqeuclidean')
        distancias[distancias == 0] = np.inf  # Evitar la diagonal (distancia a
sí mismo)
        self.sigma = np.sqrt(np.mean(np.min(distancias, axis=1)))
```

2. Centroides y Sigma

- **K-Means** identifica 10 centroides distribuidos en el espacio de los datos.
- σ se calcula como la raíz cuadrada de las distancias promedio más cercanas entre los centroides.

```
        # Paso 3: Generar la matriz de activación
        G = funcion_gaussiana(X, self.centroides, self.sigma)

# Función Gaussiana para la activación de las neuronas ocultas
def funcion_gaussiana(x, c, sigma):
    return np.exp(-cdist(x, c, 'sqeuclidean') / (2 * sigma ** 2))
```

3. Matriz de Activación

- Cada neurona oculta aplica una función Gaussiana que mide la similitud entre los datos y los centroides.

```
    def predecir(self, X):
        # Calcular la salida de la capa oculta
        G = funcion_gaussiana(X, self.centroides, self.sigma)

        # Calcular la salida final
        return np.round(G.dot(self.pesos))  # Redondear para clasificación
binaria
```

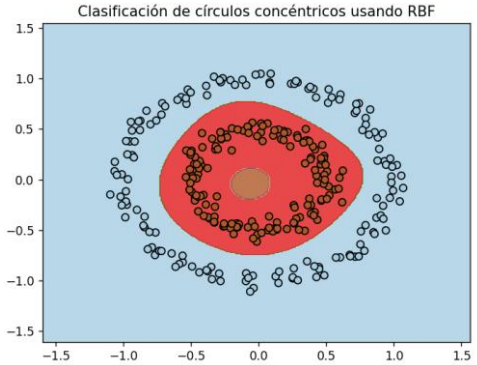
4. Predicción

- La salida final se redondea para obtener etiquetas binarias que representan las dos clases.

```
# Visualización de resultados
def graficar_resultados(X, y, modelo):
    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max,
0.01))

    Z = modelo.predecir(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.Paired)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors="k", cmap=plt.cm.Paired)
    plt.title("Clasificación de círculos concéntricos usando RBF")
    plt.show()

    # Realizar predicciones
    y_pred = rbf.predecir(X)
    print("Predicciones:", y_pred)
    print("Exactitud:", np.mean(y_pred == y))

    # Visualizar los resultados
    graficar_resultados(X, y, rbf)
```

5. Visualización

- Muestra cómo la red RBF separa las regiones correspondientes a las dos clases.



8. Conclusión

Las redes neuronales de base radial (RBF Networks) representan una solución innovadora y eficiente para abordar problemas no linealmente separables en áreas como clasificación, regresión y aproximación de funciones. Su principal fortaleza radica en la capacidad de transformar datos a espacios de mayor dimensión mediante funciones de activación basadas en la distancia radial, lo que permite una separación lineal en escenarios complejos.

La simplicidad arquitectónica de las RBF Networks, con solo una capa oculta y una capa de salida, las hace más interpretables y rápidas de entrenar en comparación con modelos como el perceptrón multicapa (MLP). Además, su flexibilidad para adaptarse a diferentes problemas y conjuntos de datos las convierte en una herramienta versátil en campos como el reconocimiento de patrones, el procesamiento de señales y el control adaptativo.

Sin embargo, el éxito de las RBF Networks depende en gran medida de la selección adecuada de parámetros clave, como los centroides y el parámetro de dispersión σ. Esto implica que el entrenamiento requiere técnicas de clustering eficientes, como K-Means, para identificar regiones significativas del espacio de datos, así como métodos robustos para calcular σ y ajustar los pesos en la capa de salida. Una mala configuración de estos parámetros puede llevar a un rendimiento deficiente, ya sea por sobreajuste o por una capacidad insuficiente para modelar los datos.

Finalmente, aunque las RBF Networks son efectivas en problemas de tamaño pequeño a mediano, su rendimiento puede degradarse con conjuntos de datos de alta dimensionalidad o tamaños masivos debido a la complejidad computacional de las distancias y las funciones radiales. Por lo tanto, su implementación exitosa requiere un balance entre simplicidad, capacidad de generalización y eficiencia computacional, lo que subraya la importancia de un diseño cuidadoso y basado en el problema específico.

En resumen, las RBF Networks son una herramienta poderosa, pero su rendimiento óptimo está condicionado por la habilidad del implementador para adaptar los parámetros y la arquitectura al contexto del problema, garantizando así su efectividad en una variedad de aplicaciones prácticas.

9. Anexos

https://github.com/EmilianoHM/Bender-IA/tree/main/Redes%20Neuronales

https://towardsdatascience.com/radial-basis-functions-neural-networks-all-we-need-to-know-9a88cc053448

https://github.com/andreiosgf/neural_nets