Radial Basis Function Networks

Maike Brochtrup, Caroline Baer

Fallstudien II, TU Dortmund

02. Dezember 2025

Radial Basis Function Networks

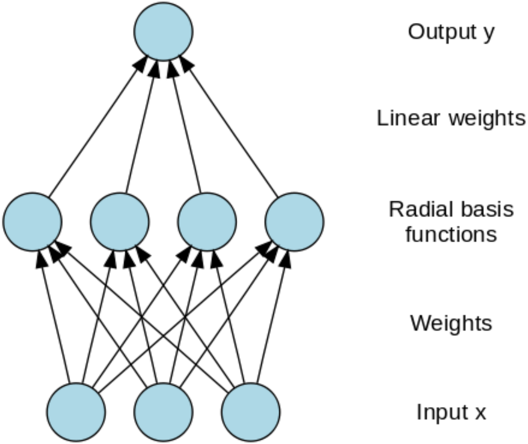
- artficial neural network
  → uses **radial basis functions** as activation functions
  ↳ symmetric function, depending only on the distance from the center point

- many applications
  → e.g. interpolation, function approximation, time series prediction and classification

- typically **3 layers**:
  1. an **input layer**
  2. a **hidden layer** with an RBF activation function
  3. and an **output layer** (linear combination of values)

- output: smooth scalar function of the inputs, interpolating all known points

Structure of the RBF Network



Mathematical Model

Output-Calculation

An RBF approximation describes the output $\hat{y}(x)$ as a weighted sum of radial functions:

$$\hat{y}(x) = \sum_{i=1}^{N} w_i \, \phi(\|x - c_i\|)$$

with:

N – number of neurons in the hidden layer

w_i – weighting parameter

$\phi(r)$ – radial function (e.g. Gaussian RBF $\phi(r) = e^{-\gamma r^2}$)

x - vector of the input parameter

c_i – center vector of neuron i

↳ often training/data points or selected by cluster methods

Advantages and Disadvantages

Advantages:

+ smooth interpolation between data points
+ flexible in non-linear contexts
+ easy to implement, visualize and interpret
+ cheaper (computing time, memory)
+ approximation if the function cannot be implemented precisely, is not known exactly, or is too expensive

Disadvantages:

- selection of suitable centers c_i
- determination of the spread parameter γ
  - ↳ strongly influences smoothing and accuracy
- no direct statistical uncertainty estimation as with kriging/Gaussian process regression

R Application

Package: RSNNS

Function: rbf

Input Variable:

- **x**: matrix with training inputs
- **y**: corresponding targets values
- **size**: number of units in the hidden layer(s)
- **maxit**: maximum of iterations to learn
- **initFunc**: initialization function to use (default is RBF_Weights)

  ⋮

- **linOut**: set the activation function of the output units to linear or logistic
- **inputsTest**: matrix with inputs to test the network
- **targetsTest**: corresponding targets fo the test input

References

Bergmeir, Christoph und José M. Benítez (2012). „Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS“. In: *Journal of Statistical Software* 46(7), S. 1–26.

Mohanty, Sthitapragyan, Prashanta Kumar Patra und Sudhansu Sekhar Sahoo (2016). „Prediction and application of solar radiation with soft computing over traditional and conventional approach – A comprehensive review“. In: *Renewable and Sustainable Energy Reviews* 56, S. 778–796. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.11.078.

Ramezanizadeh, Mahdi, Mohammad Hossein Ahmadi, Mohammad Alhuyi Nazari, Milad Sadeghzadeh und Lingen Chen (2019). „A review on the utilized machine learning approaches for modeling the dynamic viscosity of nanofluids“. In: *Renewable and Sustainable Energy Reviews* 114, S. 109345. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2019.109345.

Turkson, Richard Fiifi, Fuwu Yan, Mohamed Kamal Ahmed Ali und Jie Hu (2016). „Artificial neural network applications in the calibration of spark-ignition engines: An overview“. In: *Engineering Science and Technology, an International Journal* 19(3), S. 1346–1359. ISSN: 2215-0986. DOI: https://doi.org/10.1016/j.jestch.2016.03.003.