Aprenentatge Automàtic 1

GCED

Lluís A. Belanche

belanche@cs.upc.edu

Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)

Universitat Politècnica de Catalunya

2019-2020

LECTURE 9: Artificial Neural Networks (II)

Artificial neural networks (II): the RBF

Introduction

- RBFs have their roots at exact function interpolation (formulation as a neural network came later)

- The output of a hidden neuron is determined by the **distance** between the input and the neuron's center (seen as a **prototype**)

- This latter fact has two important consequences:

  1. It allows to give a precise interpretation to the network output

  2. It allows to design de-coupled training algorithms

Artificial neural networks (II): the RBF

Introduction

- Exact function interpolation:

$$h(\boldsymbol{x}_n) = t_n \qquad \boldsymbol{x}_n \in \mathbb{R}^d, t_n \in \mathbb{R},\ n = 1, ..., N$$

- The function h is expressed as a combination of **basis functions**:

$$\phi_n(\boldsymbol{x}) := \phi(\|\boldsymbol{x} - \boldsymbol{x}_n\|)$$

Artificial neural networks (II): the RBF

Introduction

- The combination is linear w.r.t. the basis functions:

$$h(\boldsymbol{x}) = \sum_{n=1}^{N} w_n \phi_n(\boldsymbol{x}) = \sum_{n=1}^{N} w_n \phi(\|\boldsymbol{x} - \boldsymbol{x}_n\|)$$

  which we will force to be exact for all the data points: $h(\boldsymbol{x}_n) = t_n$

- The function $\|\cdot\|$ is any norm in $\mathbb{R}^d$ (most often an **Euclidean norm**)

- Because of the norm, the ϕ_n are functions that exhibit **radial** contours of constant value **centered** at the data points $\boldsymbol{x}_n$

Artificial neural networks (II): the RBF

Introduction

In matrix notation:

$$\begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_1) & \cdots & \phi_N(\boldsymbol{x}_1) \\ \phi_1(\boldsymbol{x}_2) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \phi_2(\boldsymbol{x}_N) & \cdots & \phi_N(\boldsymbol{x}_N) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}$$

$$\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{t}$$

Note that the matrix Φ is $N \times N$ and symmetric.

Artificial neural networks (II): the RBF

Introduction

- Assuming that Φ is non-singular, $\boldsymbol{w}$ can be found as $\boldsymbol{w} = \Phi^{-1}\boldsymbol{t}$ (*e.g.*, using LU decomposition: $\Phi = LU$ where L is lower triangular, U upper triangular)

- It can be shown that indeed Φ is non-singular for various choices of the basis functions (**Micchelli's theorem**), including:

  1. $\phi(z) = \exp(-z^2/\sigma^2)$
  2. $\phi(z) = (z^2 + \sigma^2)^\alpha,\ \alpha \in (-\infty, 0) \cup (0, 1)$
  3. $\phi(z) = z^3$
  4. $\phi(z) = z^2 \ln z$

Artificial neural networks (II): the RBF

Introduction

- If the interpolation problem has codomain in $\mathbb{R}^m$ (i.e., $t_n \in \mathbb{R}^m$), the generalization is straightforward:

$$h_k(\boldsymbol{x}) = \sum_{n=1}^{N} w_{kn}\phi_n(\boldsymbol{x}) = \sum_{n=1}^{N} w_{kn}\phi(\|\boldsymbol{x} - \boldsymbol{x}_n\|), \; 1 \leq k \leq m$$

  that we will force to be exact for all the data points: $h_k(\boldsymbol{x}_n) = t_{nk}$

- This problem leads to $\Phi W = T$, solved again by simple matrix inversion as $W = \Phi^{-1}T$

Note the dimensions: Φ is $N \times N$, but W, T are $N \times m$

Artificial neural networks (II): the RBF

Regularization

- Very often, in ML, the exact function interpolation setting is not attractive at all!

  1. High number (N) of interpolation points $\rightarrow$ complex and unstable solutions

  2. The outputs t_n depend stochastically on the inputs $\boldsymbol{x}_n$ $\rightarrow$ overfit solutions

  3. The interpolation matrix Φ can be singular or ill-conditioned

  4. The inversion of Φ grows as $O(N^3)$
     (for symmetric PD matrices, Cholesky decomposition takes some $N^3/3$ steps)

- We are in need of a tighter control of complexity of the solution

Artificial neural networks (II): the RBF

Regularization

- From previous lectures, we know that **regularization** penalizes the size of the weight matrix:

$$E_{emp}(W) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{m} (t_{nk} - h_k(\boldsymbol{x}_n))^2 + \frac{\lambda}{2} \sum_{k=1}^{m} \|\boldsymbol{w}_k\|^2$$

  which results in $W = (\Phi + \lambda I_N)^{-1} T$; the value of $\lambda > 0$ is proportional to the amount of noise in the data

- Another way of obtaining much simpler solutions is to use a **subset** of the data points to center the basis functions; more generally, they can be centered at a carefully selected set of points in $\mathbb{R}^d$

Artificial neural networks (II): the RBF

RBF networks

With these modifications, we obtain the so-called RBF network:

$$h_k(\boldsymbol{x}) = \sum_{i=0}^{H} w_{ki}\phi_i(\boldsymbol{x}) = \sum_{i=0}^{H} w_{ki}\phi(\|\boldsymbol{x} - \boldsymbol{c}_i\|), \ 1 \leq k \leq m$$

which is a **two-layer neural network**:

1. The first (hidden) layer of $H \ll N$ neurons compute the basis functions $\phi_i(\boldsymbol{x})$, centered at the vectors $\boldsymbol{c}_i$

2. A constant basis function $\phi_0(\boldsymbol{x}) = 1$ compensates for the difference between the mean values of the output and the targets

Artificial neural networks (II): the RBF

RBF networks

A very popular choice for the ϕ_i is a simple Gaussian:

$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_i\|^2}{\sigma_i^2}\right)$$

- The new matrix $\Phi_{N\times(H+1)}$, is sometimes known as the **design** matrix; now the weight matrix is $W = (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}T$
- If the original $\Phi_{N\times N}$ matrix was non-singular, then the matrix $\Phi^\mathsf{T}_{N\times(H+1)}\Phi_{N\times(H+1)}$ is also non-singular (very important result!)

If we also regularize the solution, then $W = (\Phi^\mathsf{T}\Phi + \lambda I_{H+1})^{-1}\Phi^\mathsf{T}T$

Artificial neural networks (II): the RBF

In summary

RBF network training is typically performed in a decoupled way:

1. The first stage finds $H, \{\boldsymbol{c}_i\}, \{\sigma_i^2\}$ using a **clustering** algorithm

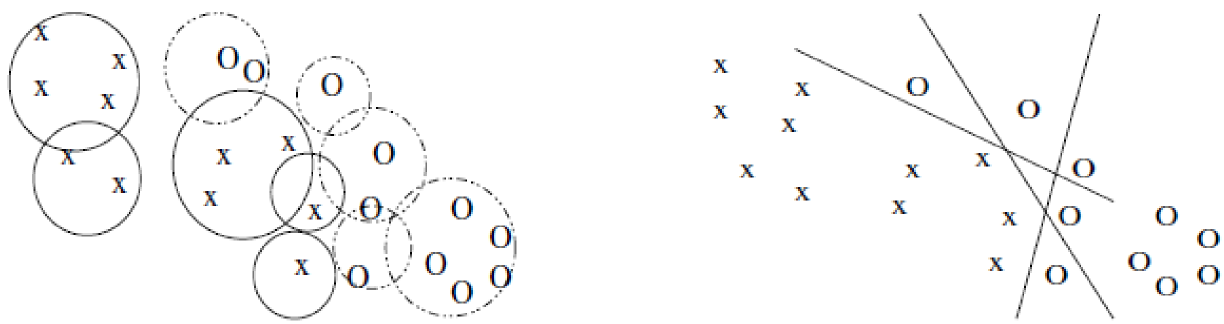
2. The second stage finds W by any of the usual (linear) methods:

   - Solution using the pseudo-inverse (via the SVD), for **regression**

   - Solution using IRLS (logistic regression), for **binary classification**

Artificial neural networks (II): the RBF

Comparison to the MLP

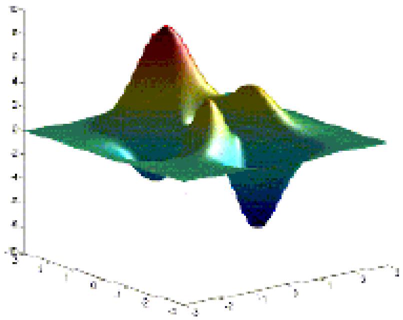


Two-class classification:

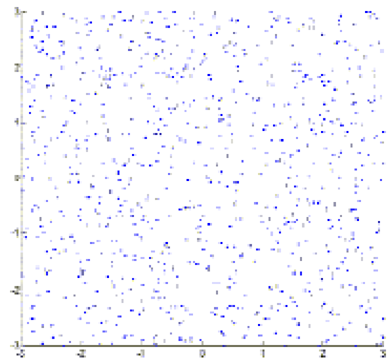
(left) separation by Gaussian neurons (RBFNN)
(right) separation by hyperplanes (MLP)

Artificial neural networks (II): the RBF

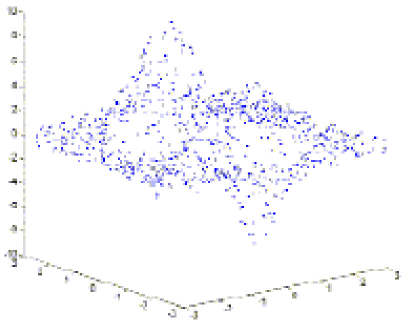
Example (I)



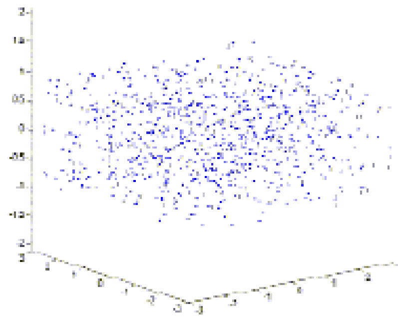
a: Deterministic function



b: Uniform distribution of data points



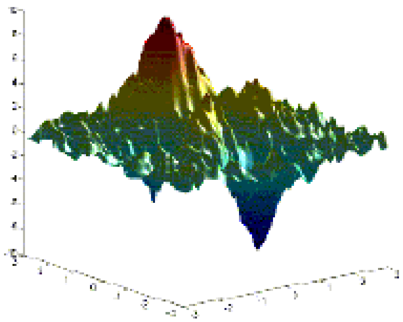
c: The data sample with noise



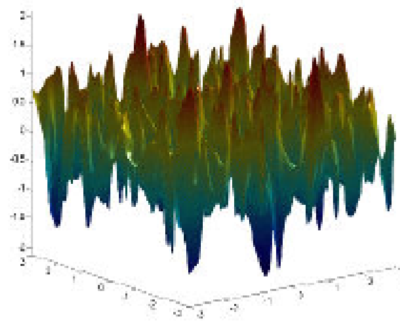
d: The U(-1,1) noise component

Artificial neural networks (II): the RBF

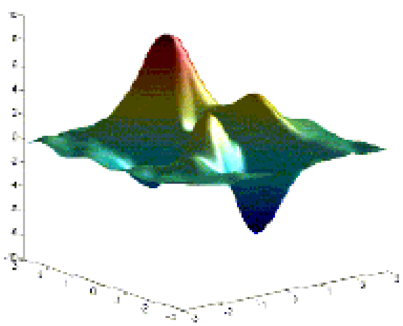
Example (II)



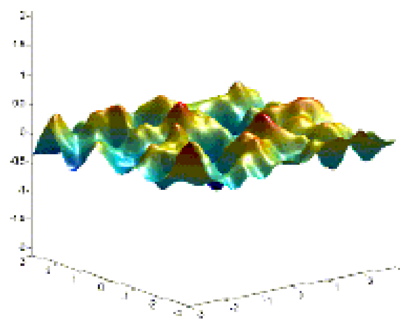
e: Exact fit to data points in (c)



f: (e)-(a), i.e., exactly fitting the data in (d)



g: Approximating RBF



h: (g)-(a)