technische universität dortmund

Computational Intelligence

Winter Term 2015/16

Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering (LS 11)

Fakultät für Informatik

TU Dortmund

Plan for Today

Lecture 03

- Application Fields of ANNs
  - Classification
  - Prediction
  - Function Approximation
- Recurrent MLP
  - Elman Nets
  - Jordan Nets
- Radial Basis Function Nets (RBF Nets)
  - Model
  - Training

Application Fields of ANNs

Lecture 03

Classification

given: set of training patterns (input / output) $\tilde{x}_i$ $\tilde{y}_i$

output = label
(e.g. class A, class B, ...)

parameters

$$f(x; (\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_m, \tilde{y}_m), w_1, \ldots, w_n) \rightarrow \hat{y}$$

input (unknown) — training patterns (known) — weights (learnt) — output (guessed)

phase I:

train network

phase II:

apply network to unkown inputs for classification

Application Fields of ANNs

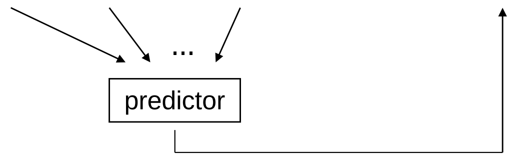
Lecture 03

Prediction of Time Series

time series $x_1, x_2, x_3, \ldots$ (e.g. temperatures, exchange rates, ...)

task: given a subset of historical data, predict the future

$$f(x_{t-k}, x_{t-k+1}, \ldots, x_t; w_1, \ldots, w_n) \rightarrow \hat{x}_{t+\tau}$$



training patterns:

historical data where true output is known;

error per pattern = $(\hat{x}_{t+\tau} - x_{t+\tau})^2$

phase I:

train network

phase II:

apply network to historical inputs for predicting <u>unkown</u> outputs

Prediction of Time Series: <u>Example for Creating Training Data</u>

given: time series 10.5, 3.4, 5.6, 2.4, 5.9, 8.4, 3.9, 4.4, 1.7

time window: k=3

(10.5, 3.4, 5.6) 2.4 first input / output pair

known input known output

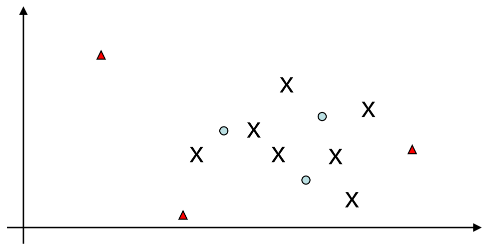
further input / output pairs:
(3.4, 5.6, 2.4) 5.9
(5.6, 2.4, 5.9) 8.4
(2.4, 5.9, 8.4) 3.9
(5.9, 8.4, 3.9) 4.4
(8.4, 3.9, 4.4) 1.7

Function Approximation (the general case)

task: given training patterns (input / output), approximate unkown function

→ should give outputs close to true unkown function for arbitrary inputs

- values between training patterns are **interpolated**
- values outside convex hull of training patterns are **extrapolated**



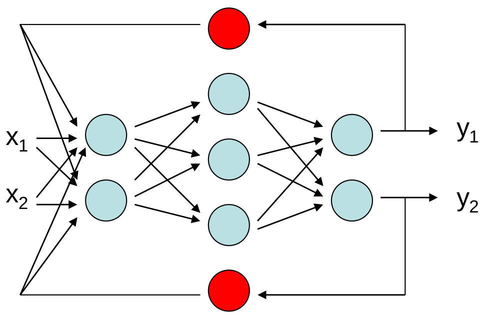
x : input training pattern

∘ : input pattern where output to be interpolated

▲ : input pattern where output to be extrapolated

Jordan nets (1986)

- **context neuron**:
  reads output from some neuron at step t and feeds value into net at step t+1



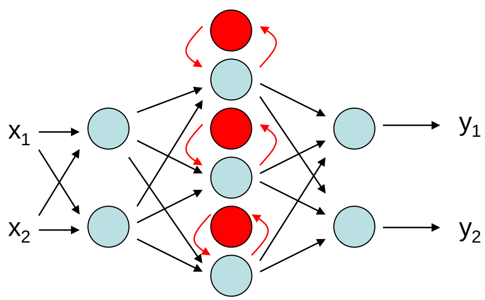
Jordan net =

MLP + context neuron for each output, context neurons fully connected to input layer

Elman nets (1990)

Elman net =

MLP + context neuron for each hidden layer neuron‘s output of MLP, context neurons fully connected to emitting MLP layer



Training?

⇒ unfolding in time ("loop unrolling")

- identical MLPs serially connected (finitely often)
- results in a large MLP with many hidden (inner) layers
- backpropagation may take a long time
- but reasonable if most recent past more important than layers far away

Why using backpropagation?

⇒ use *Evolutionary Algorithms* directly on recurrent MLP!

later!

Definition:
A function $\phi : \mathbb{R}^n \to \mathbb{R}$ is termed **radial basis function**
iff $\exists \varphi : \mathbb{R} \to \mathbb{R} : \forall x \in \mathbb{R}^n : \phi(x; c) = \varphi(\,\|x - c\|\,)$. □

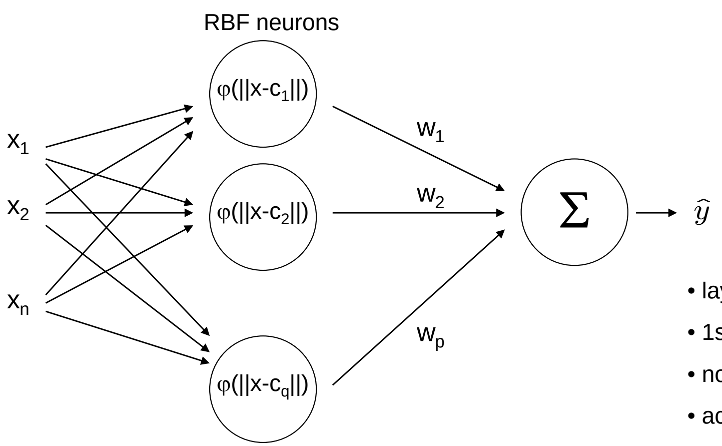
Definition:
RBF **local** iff
$\varphi(r) \to 0$ as $r \to \infty$ □

typically, || x || denotes Euclidean norm of vector x

examples:

$\varphi(r) = \exp\left(-\frac{r^2}{\sigma^2}\right)$	Gaussian	unbounded	local
$\varphi(r) = \frac{3}{4}(1 - r^2) \cdot 1_{\{r \le 1\}}$	Epanechnikov	bounded	local
$\varphi(r) = \frac{\pi}{4}\cos\left(\frac{\pi}{2} r\right) \cdot 1_{\{r \le 1\}}$	Cosine	bounded	local

Definition:
A function f: $\mathbb{R}^n \to \mathbb{R}$ is termed **radial basis function net (RBF net)**
iff $f(x) = w_1\, \varphi(\| x - c_1 \|) + w_2\, \varphi(\| x - c_2 \|) + \ldots + w_p\, \varphi(\| x - c_q \|)$ □



- layered net
- 1st layer fully connected
- no weights in 1st layer
- activation functions differ

given : N training patterns (x_i, y_i) and q RBF neurons

find : weights $w_1, \ldots, w_q$ with minimal error

solution:

we know that $f(x_i) = y_i$ for i = 1, ..., N and therefore we insist that

$$\underbrace{\sum_{k=1}^{q} w_k}_{\text{unknown}} \cdot \underbrace{\varphi(\|x_i - c_k\|)}_{p_{ik}\text{: known value}} = \underbrace{y_i}_{\text{known value}}$$

$$\Rightarrow \sum_{k=1}^{q} w_k \cdot p_{ik} = y_i$$

⇒ N linear equations with q unknowns

in matrix form: P w = y with P = (p_{ik}) and P: N x q, y: N x 1, w: q x 1,

case N = q: w = P^{-1} y if P has full rank

case N < q: many solutions but of no practical relevance

case N > q: w = P^{+} y where P^{+} is Moore-Penrose pseudo inverse

P w = y | · P' from left hand side (P' is transpose of P)

P'P w = P' y | · $(P'P)^{-1}$ from left hand side

$\underbrace{(P'P)^{-1} P'P}_{\text{unit matrix}}$ w = $\underbrace{(P'P)^{-1} P'}_{P^+}$ y | simplify

- existence of $(P'P)^{-1}$?
- numerical stability ?

Tikhonov Regularization (1963)

idea:
choose $(P'P + h\, I_q)^{-1}$ instead of $(P'P)^{-1}$ ($h > 0$, I_q is q-dim. unit matrix)

excursion to linear algebra:

Def : matrix A positive semidefinite (p.s.d) iff $\forall x \in \mathbb{R}^n : x'Ax \geq 0$
Def : matrix A positive definite (p.d.) iff $\forall x \in \mathbb{R}^n \setminus \{0\} : x'Ax > 0$
Thm : matrix $A : n \times n$ regular $\Leftrightarrow$ rank$(A) = n \Leftrightarrow A^{-1}$ exists $\Leftarrow A$ is p.d.

Lemma : $a, b > 0,\ A, B : n \times n,\ A$ p.d. and B p.s.d. $\Rightarrow a \cdot A + b \cdot B$ p.d.

Proof : $\forall x \in \mathbb{R}^n \setminus \{0\} : x'(a \cdot A + b \cdot B)x = \underbrace{a}_{>0} \cdot \underbrace{x'Ax}_{>0} + \underbrace{b}_{>0} \cdot \underbrace{x'Bx}_{\geq 0} > 0$ q.e.d.

Lemma : $P : n \times q \Rightarrow P'P$ p.s.d.

Proof : $\forall x \in \mathbb{R}^n : x'(P'P)x = (x'P') \cdot (Px) = (Px)'(Px) = \|Px\|_2^2 \geq 0$ q.e.d.

Tikhonov Regularization (1963)

$\Rightarrow (P'P + h\, I_q)$ is p.d. $\Rightarrow (P'P + h\, I_q)^{-1}$ exists

question: how to justify this particular choice?

$\|Pw - y\|^2 + h \cdot \|w\|^2 \quad \to \min_w!$

interpretation: minimize TSSE and prefer solutions with small values!

$\frac{d}{dw}[\,(Pw - y)'(Pw - y) + h \cdot w'w\,] =$

$\frac{d}{dw}[\,(w'P'Pw - w'P'y - y'Pw + y'y + h \cdot w'w\,] =$

$2\,P'Pw - 2\,P'y + 2h\,w = 2\,(P'P + h\,I_q)\,w - 2\,P'y \overset{!}{=} 0$

$\Rightarrow w^* = (P'P + h\,I_q)^{-1}P'y$

$\frac{d}{dw}[\,2\,(P'P + h\,I_q)\,x - 2\,P'y\,] = 2\,(P'P + h\,I_q)$ is p.d. $\Rightarrow$ minimum

Tikhonov Regularization (1963)

question: how to find appropriate $h > 0$ in $(P'P + h\,I_q)$?

let $\text{PERF}(h; T)$ with $\text{PERF} : \mathbb{R}^+ \to \mathbb{R}^+$ measure the performance of RBF net for positive h and given training set T

find h^* such that $\text{PERF}(h^*; T) = \max\{\text{PERF}(h; T) : h \in \mathbb{R}^+\}$

→ several approaches in use
→ here: **grid search** and **crossvalidation**

(1) choose $n \in \mathbb{N}$ and $h_1, \ldots, h_n \in (0, H] \subset \mathbb{R}^+$; set $p^* = 0$
(2) for $i = 1$ to n
(3) $p_i = \text{PERF}(h_i; T)$
(4) if $p_i > p^*$
(5) $p^* = p_i; k = i;$
(6) endif
(7) endfor
(8) return h_k

} grid search

Crossvalidation

choose $k \in \mathbb{N}$ with $k < |T|$
let $T_1, \ldots, T_k$ be partition of training set T

$T_1 \cup \ldots \cup T_k = T$
$T_i \cap T_j = \emptyset$ for $i \neq j$

PERF$(h; T) =$
(1) set $err = 0$
(2) `for` $i = 1$ `to` k
(3) build matrix P and vector y from $T \setminus T_i$
(4) get weights $w = (P'P + h\,I)^{-1} P'y$
(5) build matrix P and vector y from T_i
(6) get error $e = (Pw - y)'(Pw - y)$
(7) $err = err + e$
(8) `endfor`
(9) `return` $1/err$

complexity (naive)

w = (P'P) $^{-1}$ P' y

P'P: N^2 q inversion: q^3 P'y: qN multiplication: q^2

O(N^2 q) elementary operations

remark: if N large then inaccuracies for P'P likely

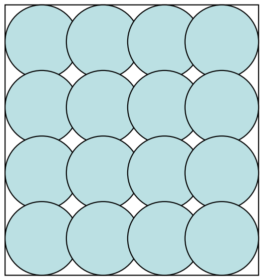
⇒ first analytic solution, then gradient descent starting from this solution

requires differentiable basis functions!

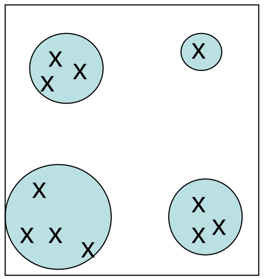
so far: tacitly assumed that RBF neurons are given

⇒ center c_k and radii σ considered given and known

how to choose c_k and σ ?



uniform covering



if training patterns inhomogenously distributed then first cluster analysis

choose center of basis function from each cluster, use cluster size for setting σ

advantages:

- additional training patterns → only local adjustment of weights
- optimal weights determinable in polynomial time
- regions not supported by RBF net can be identified by zero outputs

(if output close to zero, verify that output of each basis function is close to zero)

disadvantages:

- number of neurons increases exponentially with input dimension
- unable to extrapolate (since there are no centers and RBFs are local)