Hybrid learning methods in Bang and Regularization Networks

P. Kudová

Department of Theoretical Computer Science
Institute of Computer Science
Academy of Sciences of the Czech Republic

Edinburgh – June 2, 2005

Outline

- Bang project overview
  - MAS for experiments with AI methods
  - very brief introduction to Bang project
- Hybrid learning for RBF networks
  - introduction to RBF networks
  - introduction to learning methods for RBF
  - implementation in Bang
  - experimental results
- Regularization Networks
  - feed-forward networks related to RBF
  - derivation from Regularization Theory
  - choice of parameters (type of kernel, reg. parameter)
  - special types of kernels - Sum and Product Kernels
  - experimental results

Bang project http://bang.sf.net

- a project developed at the Institute of Computer Science
- by a group of Phd students under supervison of R. Neruda
- Bang was originaly abreviation
- What is Bang? Official short answer:

Bang is a multi-agent system (MAS) intended primarily for experimenting with computational intelligence models. It is a distributed, multiprocess/multi-thread, user-friendly, modular environment allowing for data-driven hybrid modelling with components like artificial neural networks, genetic algorithms, etc. As other MASes, Bang consists of environment and agents that communicate via messages.

Motivation

- soft computing, neural networks, genetic algorithms
- we believe in hybrid models, i.e. combination of different approaches
- inteligent framework above individual methods

- Goals:
  - environment for easy creation of hybrid models
  - encapsulation of individual methods into building blocks
  - intelligent and autonomous system
  - autonomous decisions about choice of method, based on previous knowledge

Implementation by software agents

S. Franklin: *An agent is "An autonomous a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."*

Bang agents are running entities capable to communicate (sending and receiving messages, based on ACL)

Environment is a service layer providing all the necessities

Distributed environment can be created by several network connected airports

Agent examples decision tree, neural network, Yellow Pages

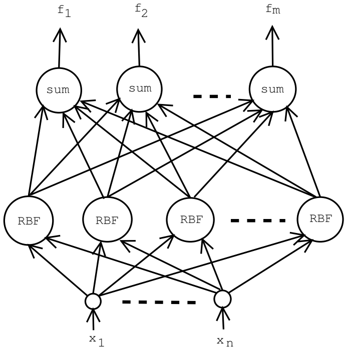
RBF networks in Bang

- neural network - black box with *n* inputs and *m* outputs
- ability to learn (usually from examples)

- very popular and widely used model of neural network
- a wide range of learning algorithms exists
- three main learning approaches implemented in Bang
- experimental study of possible combination of these methods

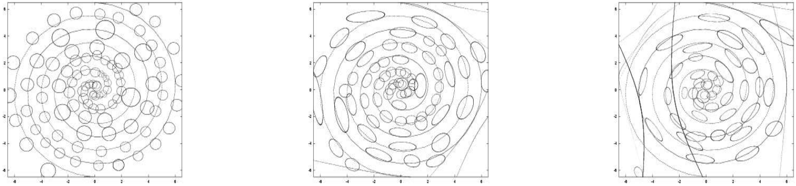
RBF networks

- One hidden layer of RBF units
  $y(\vec{x}) = \varphi\left(\frac{\|\vec{x}-\vec{c}\|_C}{b}\right)$
- Linear output layer
- Net function
  $f_s(\vec{x}) = \sum_{j=1}^{h} w_{js}\varphi\left(\frac{\|\vec{x}-\vec{c}_j\|_C}{b_j}\right)$
- $\| \vec{x} \|_C = (C\vec{x})^T(C\vec{x}) = \vec{x}^T C^T C\vec{x}$

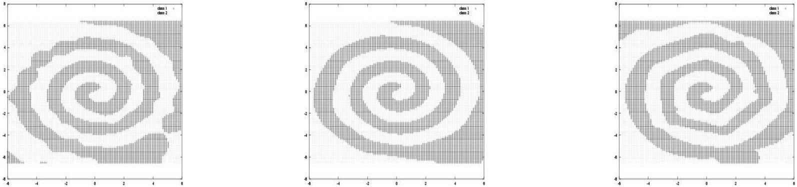


RBF units with weighted norm

100 radial | 70 oval | 50 oval



Resulting classification



Learning methods for RBF networks

- Goal of learning?
  - to approximate a given function
  - using set of examples – **training set**:
    $T = \{(\vec{x}^t, \vec{y}^t), t = 1, ..., k\}$

- How we meassure the quality of our network?
  - use appropriate *error function*
  - sum of square errors
    $E = \frac{1}{2} \sum_{t=1}^{k} \parallel \vec{y}^{(t)} - \vec{f}^{(t)} \parallel^2$

- How the network learns?
  - minimize the error function, wide range of methods
  - Gradient methods, Three-step approaches, Evolutionary methods

Gradient learning of RBF networks

- inspired by Back Propagation for MLP perceptron
- minization of the error function by gradient algorithm
- easy evaluation of derivatives, only one hidden layer
- all parameters are treated in the same way
- iterative algorithm

- control of overfiting:
  - use evalution set
  - stop when error on evaluation set starts increasing

Three step learning

- three groups of parameters
- use knowledge about character of these parameters

First step set centers of RBF units

- approximate the distribution of training samples
- random or uniform samples
- various clustering methods (k-means)

Second step set widths, norm matrices

- cover the input space by unit's fields
- heuristics (k-neighbours)

Third step set weights of hidden layer

- linear system, pseudoinverse

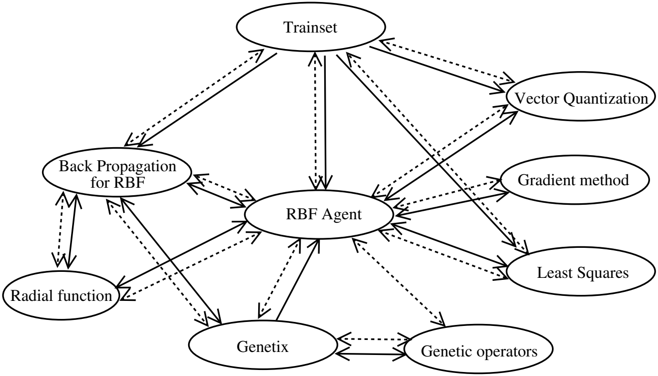
Genetic learning

- stochastic search based on evolutionary principles
- working with population of individuals, each individual represents one encoded feasible solution
- creating new individuals by means of selection, crossover and mutation
- cannonical version

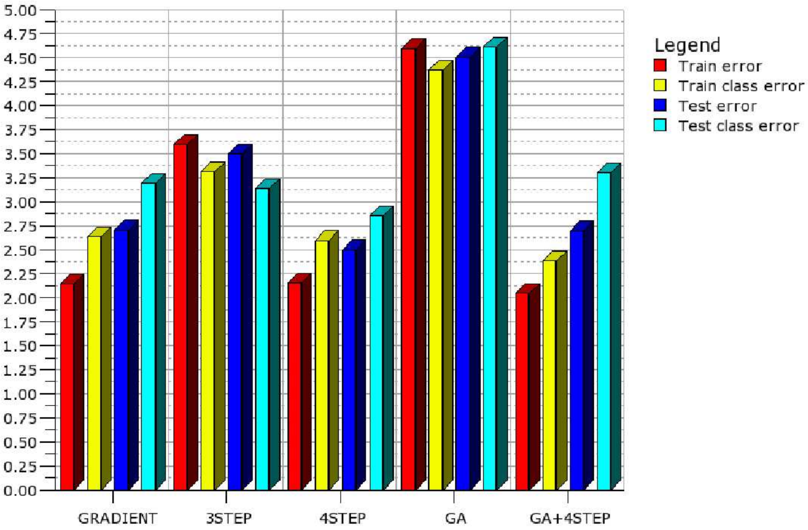
$$\boxed{\boxed{B_1}\;\boxed{B_2}\;\cdots\;\boxed{B_i}\;\cdots\;\boxed{B_h}}$$

Goal	**Block**
All networks parametres	$\vec{c}_i, b_i, C_i, w_i$
Hidden layer parametres	$\vec{c}_i, b_i, C_i$
RBF centres	$\vec{c}_i$

RBF networks in Bang



Experimental results - RBF networks



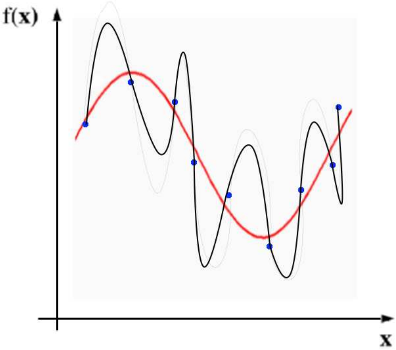
Regularization Theory & Regularization Networks

- **Regularization Networks** - a family of feedforward neural networks with one hidden layer
- derived from approximation theory
- very good theoretical background (Smola, Poggio, Shoelkopf)
- belongs to *kernel methods*

- we are interested in their real applicability

Learning from examples - problem statement

- **Given:** set of data samples $\{(\vec{x}_i, y_i) \in R^d \times R\}_{i=1}^N$
- **Our goal:** recover the unknown function or find the best estimate of it



Regularization Theory

Empirical Risk Minimization:

- find f that minimizes $H[f] = \frac{1}{N}\sum_{i=1}^{N}(f(\vec{x}_i) - y_i)^2$
- generally ill-posed
- choose one solution according to <u>a priori knowledge</u> (*smoothness, etc.*)

Regularization approach

- add a **stabiliser** $H[f] = \frac{1}{N}\sum_{i=1}^{N}(f(\vec{x}_i) - y_i)^2 + \gamma\Phi[f]$

Reproducing Kernel Hilbert Space

Definition and properties

- RKHS is a Hilbert space of functions defined over $\Omega \subset \Re^d$ with the property that for each $x \in \Omega$ the evaluation functional on $\mathcal{H}$ given by $\mathcal{F}_x : f \rightarrow f(x)$ is bounded. *(Aronszajn, 1950)*

- This implies existence of positive definite symmetric function $K : \Omega \times \Omega \rightarrow \Re$ (*kernel function*) such that

$$\mathcal{H} = \mathcal{H}_K = \mathtt{comp}\{\sum_{i=1}^{n} a_i K_{x_i};\ x_i \in \Omega, a_i \in \Re\},$$

where $\mathtt{comp}$ means completion of the set.

Reproducing Kernel Hilbert Space

Application in learning

- Data set: $\{(\vec{x}_i, y_i) \in R^d \times R\}_{i=1}^N$
- choose a symmetric, positive-definite kernel $K = K(\vec{x}_1, \vec{x}_2)$
- let $\mathcal{H}_K$ be the RKHS defined by K
- define the stabiliser by the norm $||\cdot||_K$ in $\mathcal{H}_K$

$$H[f] = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2 + \gamma||f||_K^2$$

- minimise $H[f]$ over $\mathcal{H}_K$ $\longrightarrow$ solution:

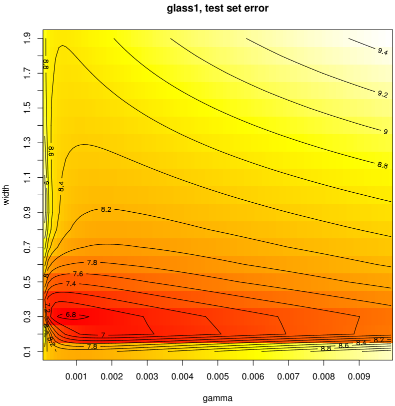
$$f(\vec{x}) = \sum_{i=1}^{N} c_i K_{\vec{x}_i}(\vec{x}) \qquad (N\gamma I + K)\vec{c} = \vec{y}$$

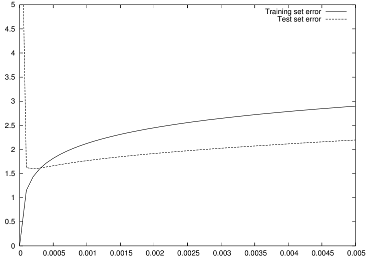
RN learning algorithm

- algorithm is quite simple, reduces to solving a linear system
$$(N\gamma I + K)\vec{c} = \vec{y}$$
- $N\gamma I + K$ strictly positive $\rightarrow$ system is **well-posed**
- Is it also **well-conditioned** ?

- For large $\gamma \rightarrow$ dominant diagonal $\rightarrow$ good
- ☹ choice of γ is not free
- ☺ using Gaussian kernel allows choice of width
- choice of γ and width is crucial for the performance of the algorithm (cross-validation)

Choice of γ and type of kernel





epcc

Model selection

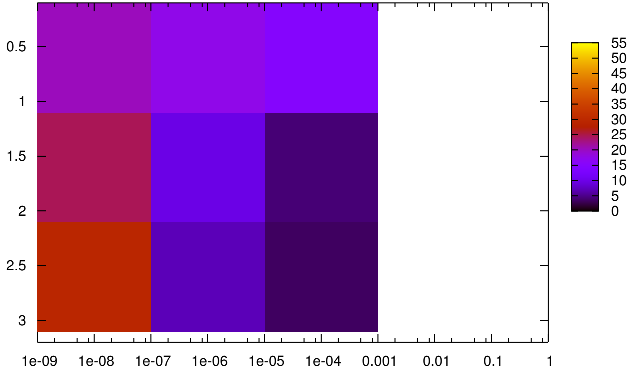
Parameters of the basic algorithm

- kernel type
- kernel parameter(s) (i.e. width for Gaussian)
- regularization parameter γ

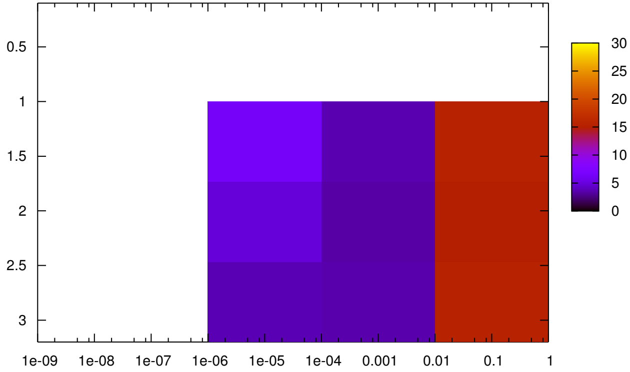
How we estimate these parameters?

- kernel type by user
- kernel parameter and regularization parameter by grid search and crossvalidation
- speed-up techniques: grid refining

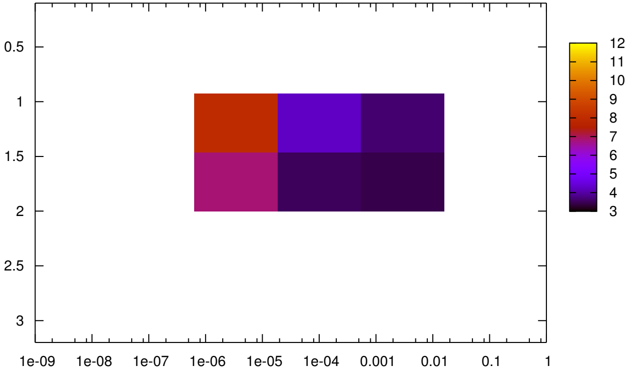
Parameter search



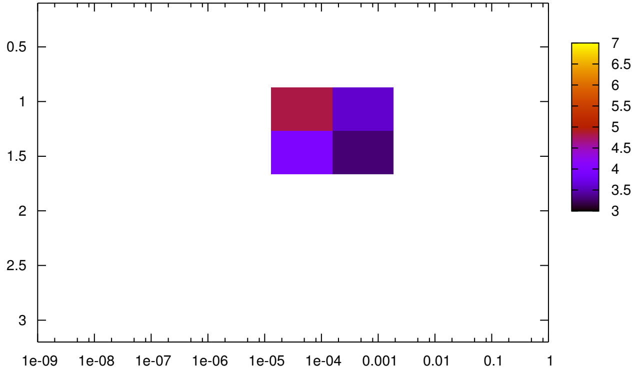
Parameter search



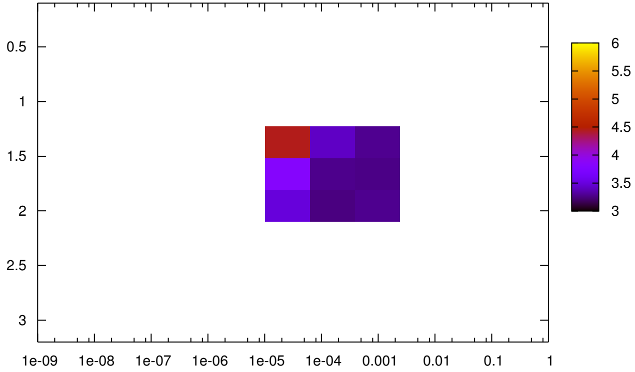
Parameter search



Parameter search



Parameter search



Grid parameter search and crossvalidation

- time consuming, many evaluations
- danger of overfitting (with respect to a particual partitioning to folds)

Work in progress

- applaying simple genetic algorithm (include choice of a kernel type)
- random partitioning in crossvalidation
- lazy evaluations

RN versus RBF networks

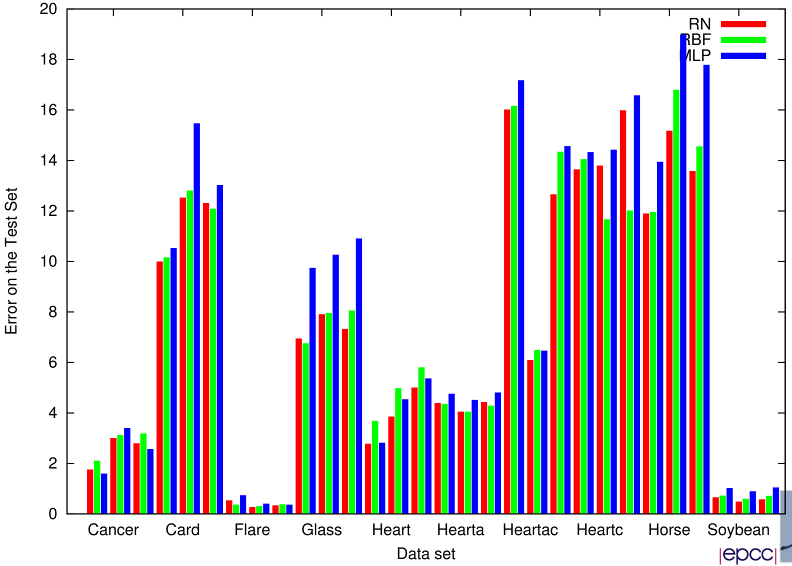
Regularization Networks

- good theoretical background, search for optimal solution
- solving linear systems by numerical algorithms
- network complexity (number of parameters) depends on the size of the training set
- parameters (γ, width)

RBF networks

- search for approximate solution (lower number of hidden units)
- algorithms for optimisation, heuristics
- network complexity does not depend on the size of the train. set, but units have more parameters $(n+1+n(n+1)/2)$
- parameter h

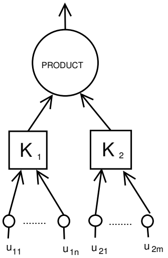
Comparison of RN and RBF on Proben1 repository

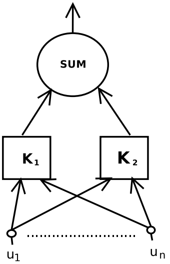


epcc

Product and Sum Kernels

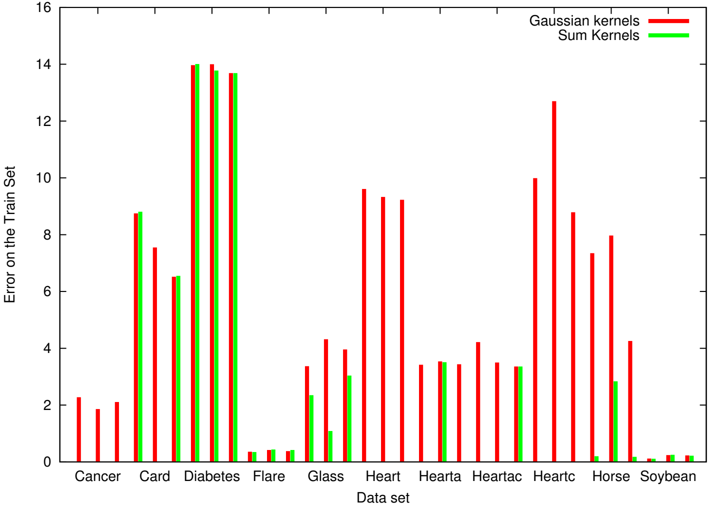
- choice of kernels depands on data, attributes types
- sometime data are not homogenous
- composite kernels - product and sum kernels may better reflect the character of data (joint work with T. Samalova)





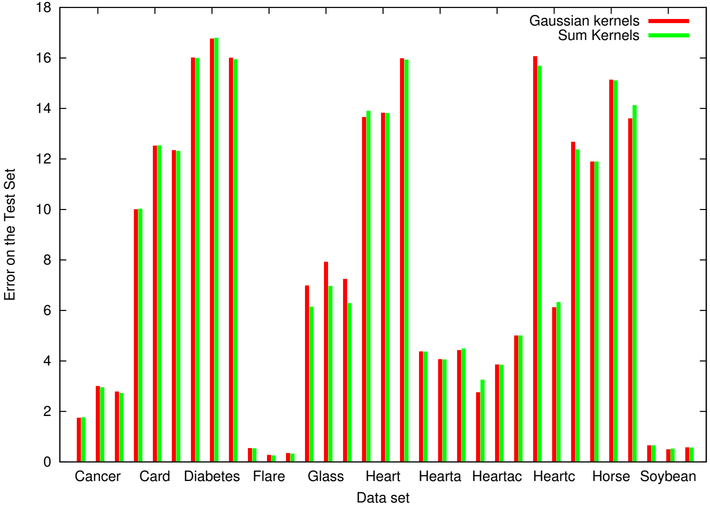
Sum versus Gaussian Kernels

The error on the training set



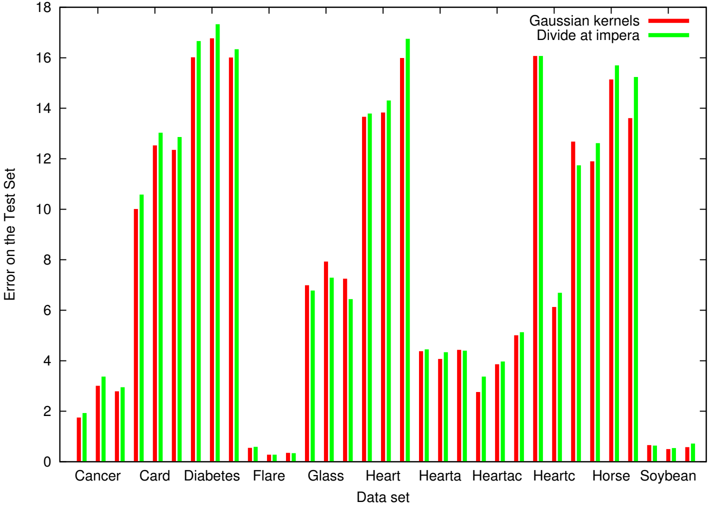
Sum versus Gaussian Kernels

The error on the testing set



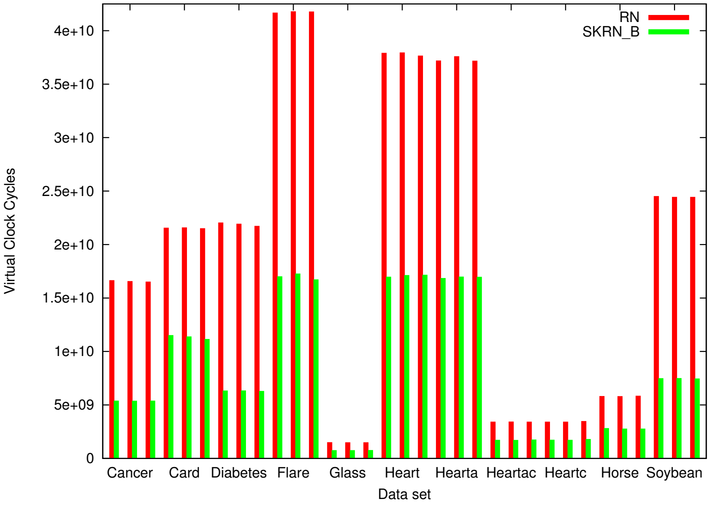
Big task? Divide and conquer!

The error on the testing set



Big task? Divide and conquer!

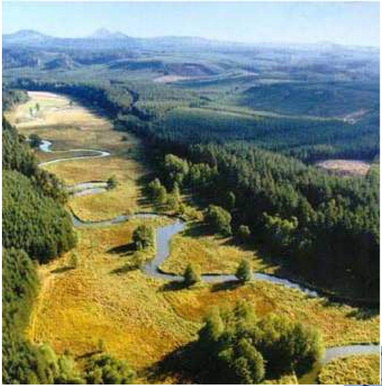
Time comparison



Prediction of flow rate

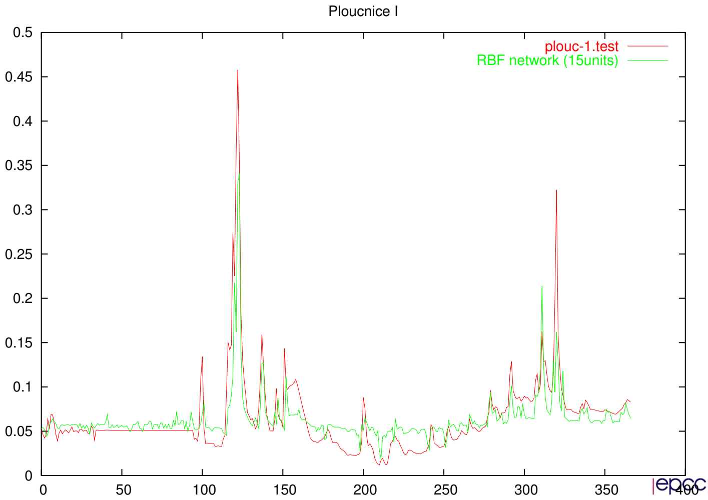
- prediction of the flow rate on the Ploučnice in North Bohemia, from origin (southwest part of the Ještěd hill) to the town Mimoň

- time series containing daily flow and rainfall values
- prediction of the current flow rate based on information from the previous one or two days
- 1000 training samples, 367 testing samples



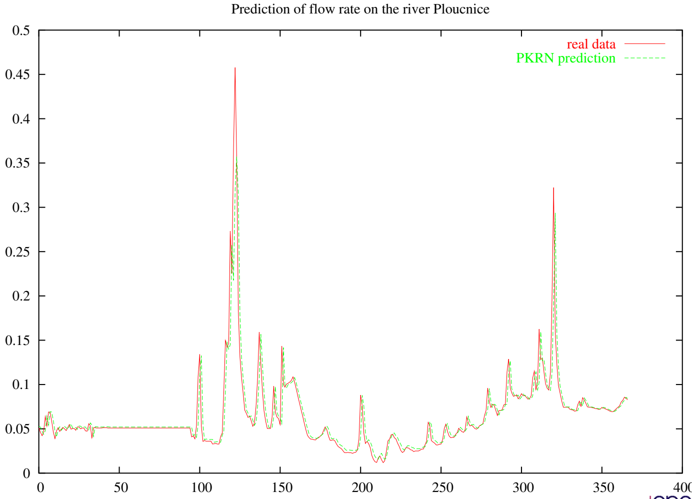
Prediction of flow rate

Prediction by RBF network



Prediction of flow rate

Prediction by Product Kernels



Summary and future work

Summary

- Bang project was introduced
- hybrid learning of RBF networks
- learning Regularization Networks, special kernel types

Work in progress and future work

- improve the parameter search including the selection of kernel type
- parameter search using evolutionary approaches