LEARNING WITH KERNEL BASED REGULARIZATION NETWORKS

P. Kudová

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

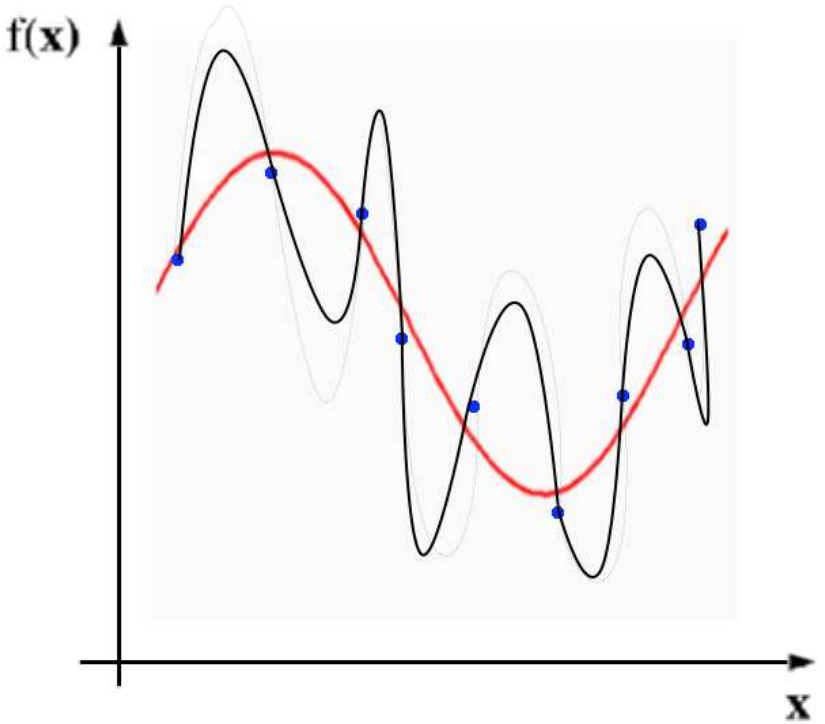
ITAT 2005, Roháče, Slovakia

Outline

- Introduction
  - Learning from examples
- Regularization Networks
  - regularization theory
  - learning algorithm
- Estimation of regularization parameter and kernel type
  - crossvalidation
  - adaptive grid search
  - genetic parameter search
- Experiments
  - composite kernels
  - comparison of different types of kernels

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



Examples of learning techniques

- Artificial neural networks
  - biological motivations
  - black box approach
  - multilayer perceptrons, RBF networks, etc.
- Decision trees, Support vector machines, etc.
- Approximation theory
  - function approximation
  - regularization theory

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 a priori knowledge (*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

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

where comp means completion of the set.

RKHS and 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 $\mathcal{H}_K \quad \longrightarrow \quad$ solution:

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

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** ?

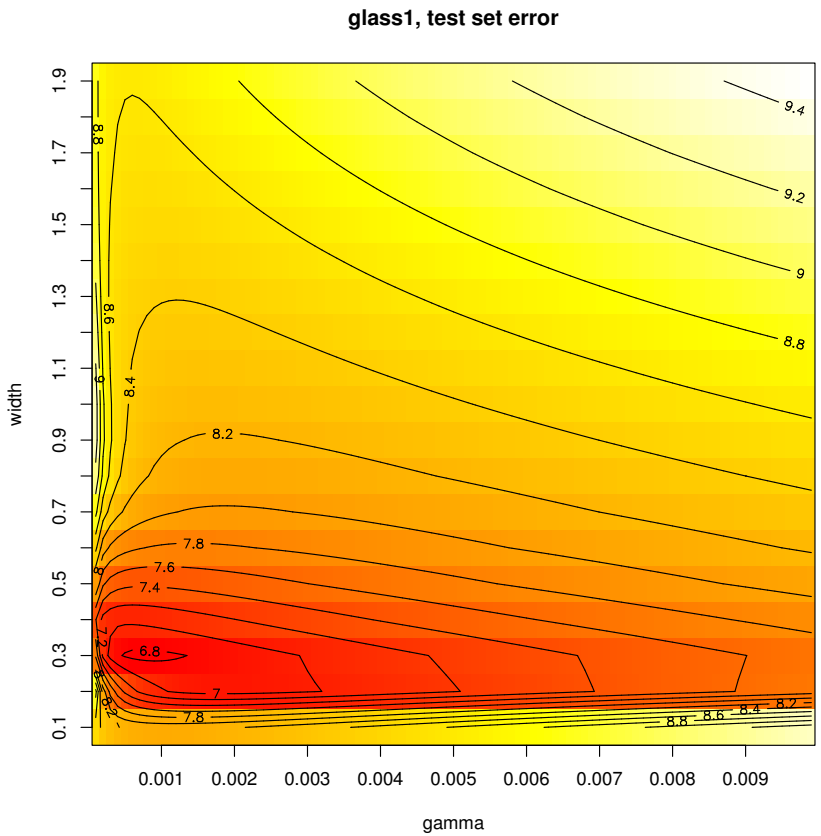
- For large $\gamma \rightarrow$ dominant diagonal $\rightarrow$ good

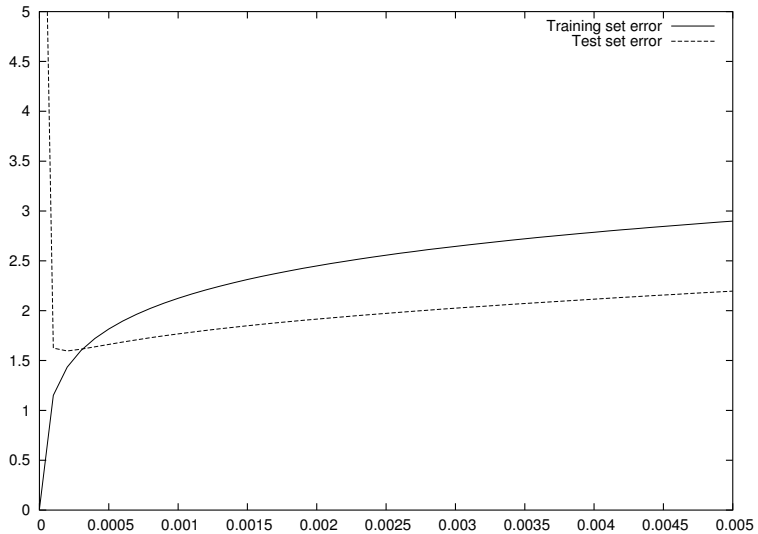

choice of γ is not free


choice of kernel and its parameters

- choice of γ and kernel type is crucial for the performance of the algorithm

Choice of γ and type of kernel





Model selection

Parameters of the basic algorithm

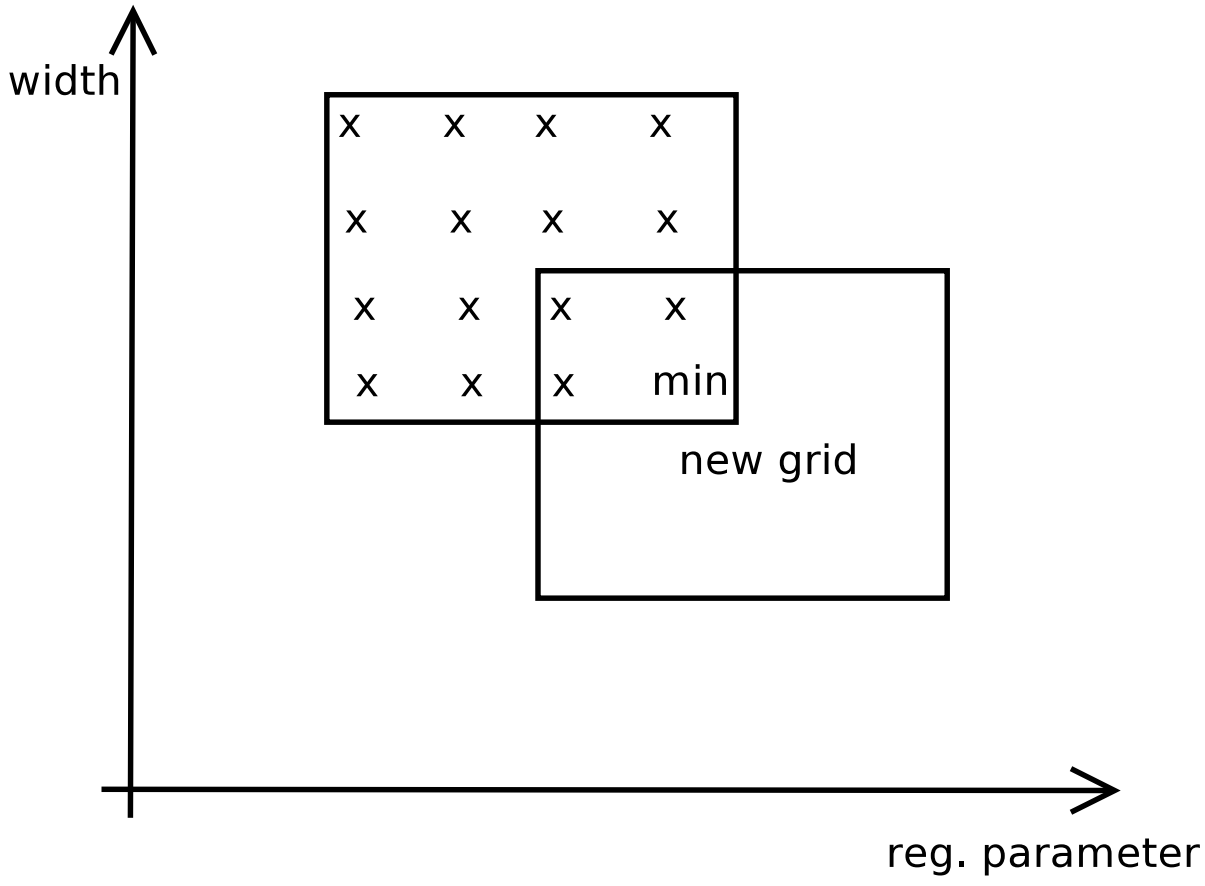
- kernel type
- kernel parameter(s) (i.e. width for Gaussian)
- regularization parameter γ

How we estimate these parameters?

- kernel type by user
- kernel's parameters and regularization parameter - search for parameters with the lowest cross-validation error
- grid search, genetic search

Parameter search - Adaptive Grid

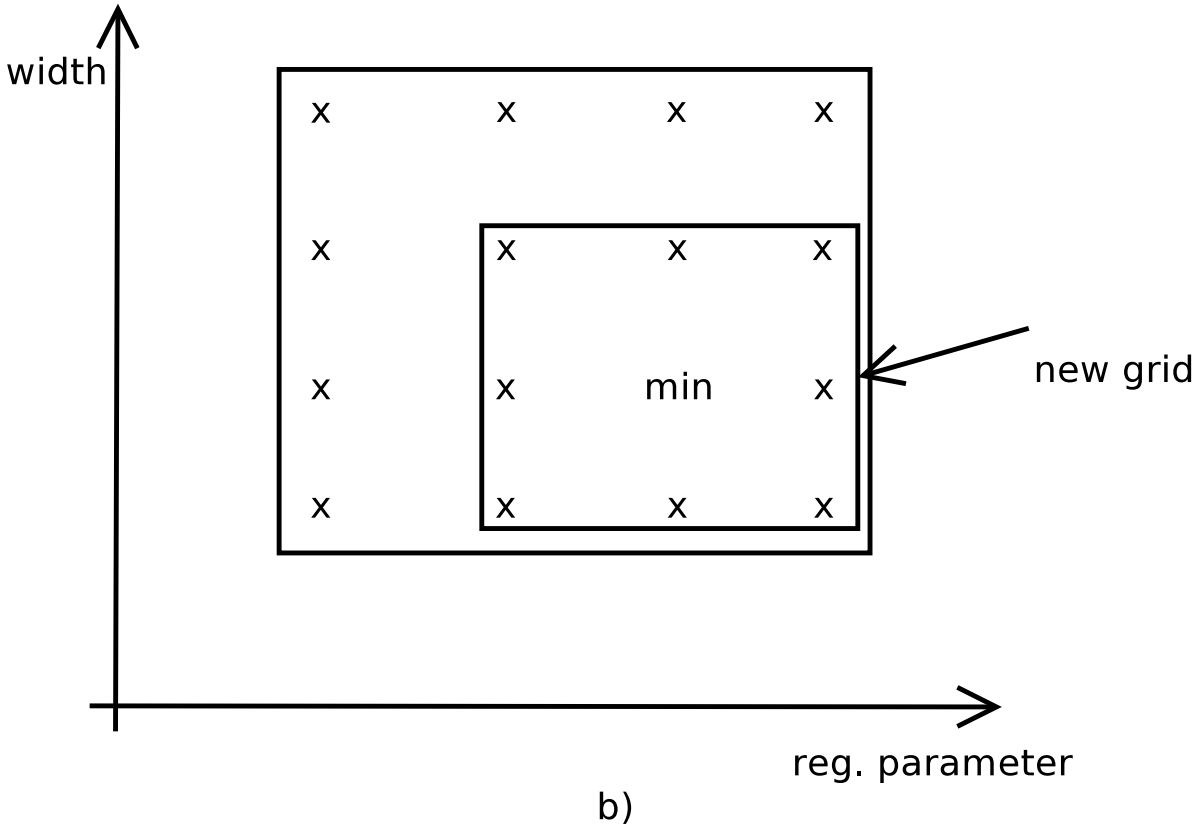
Move grid



a)

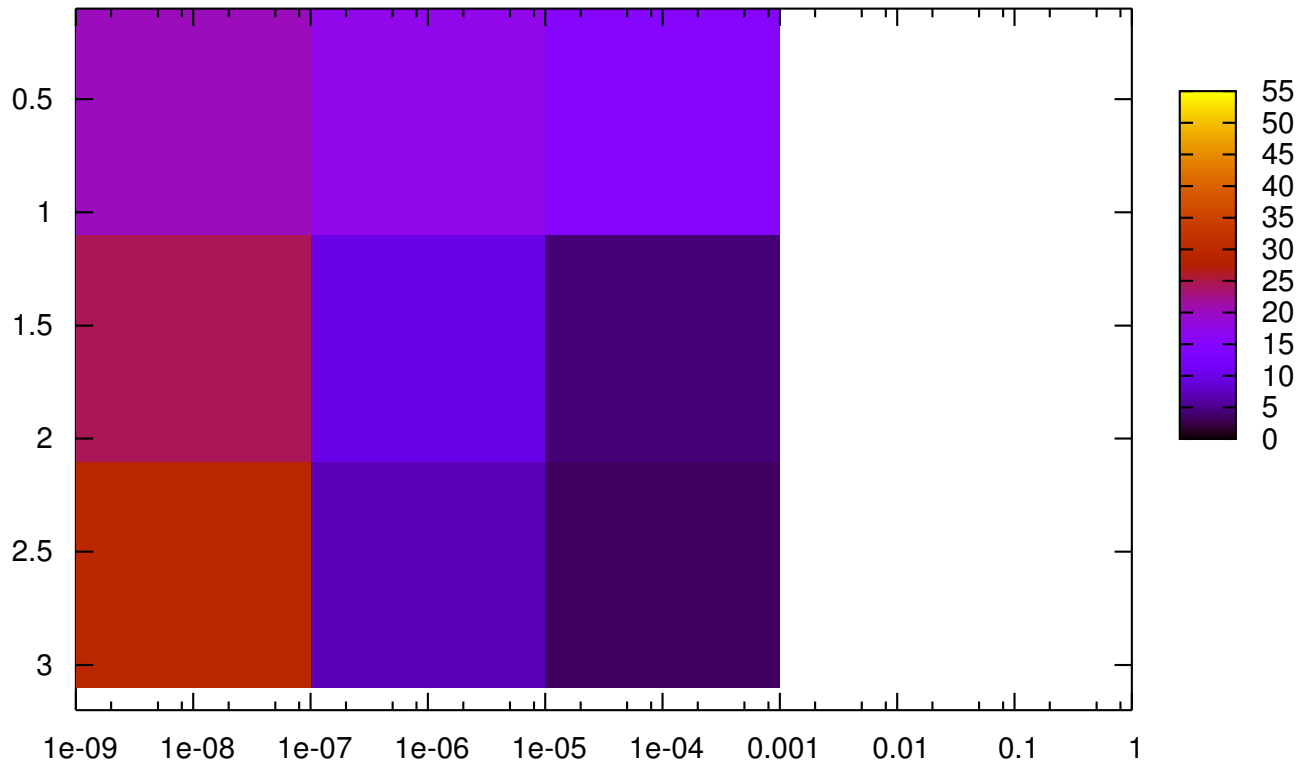
Parameter search - Adaptive Grid

Refine grid

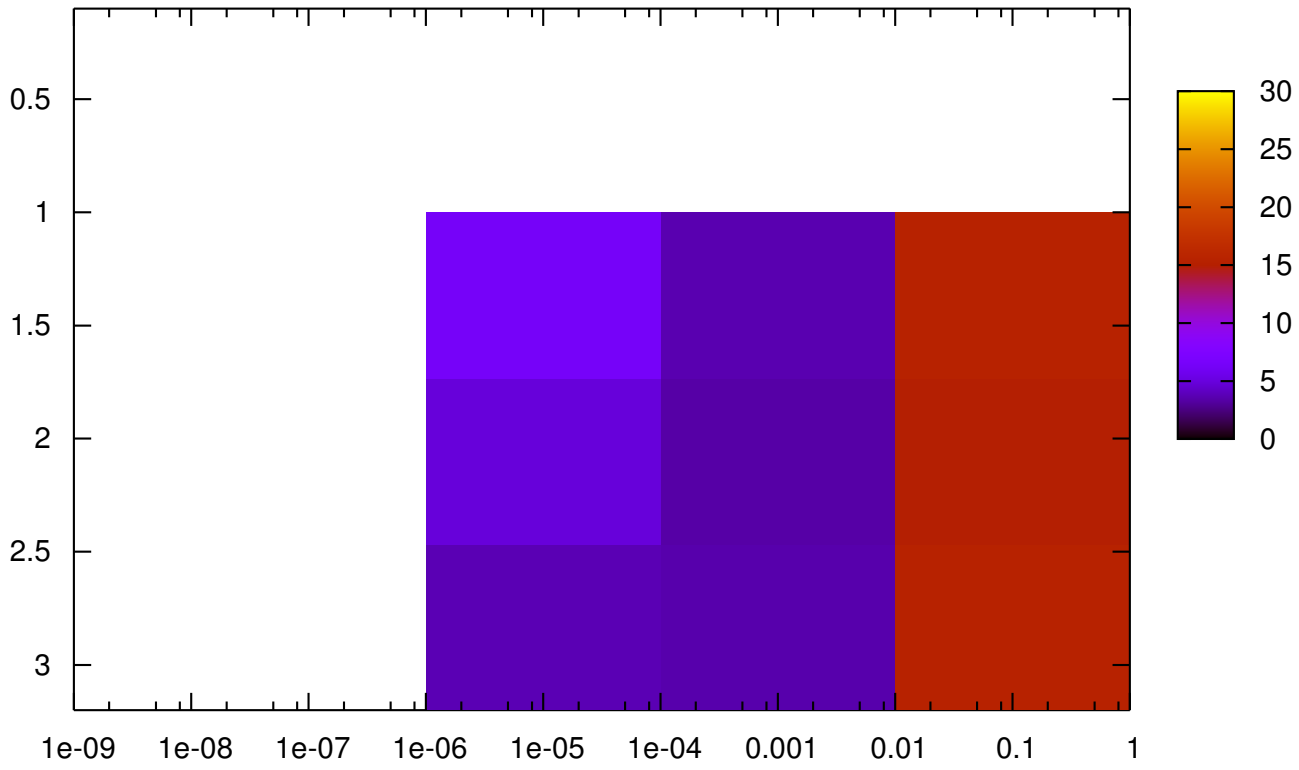


b)

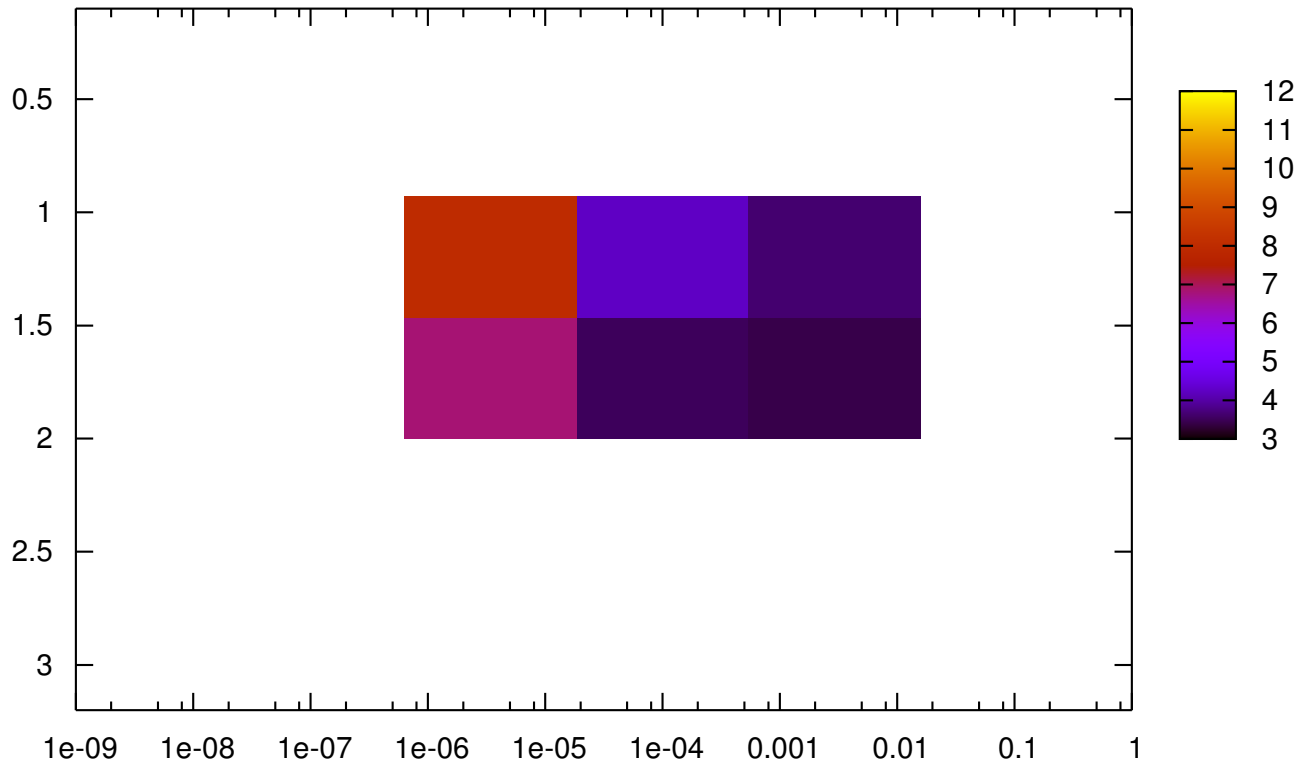
Parameter search - Adaptive Grid



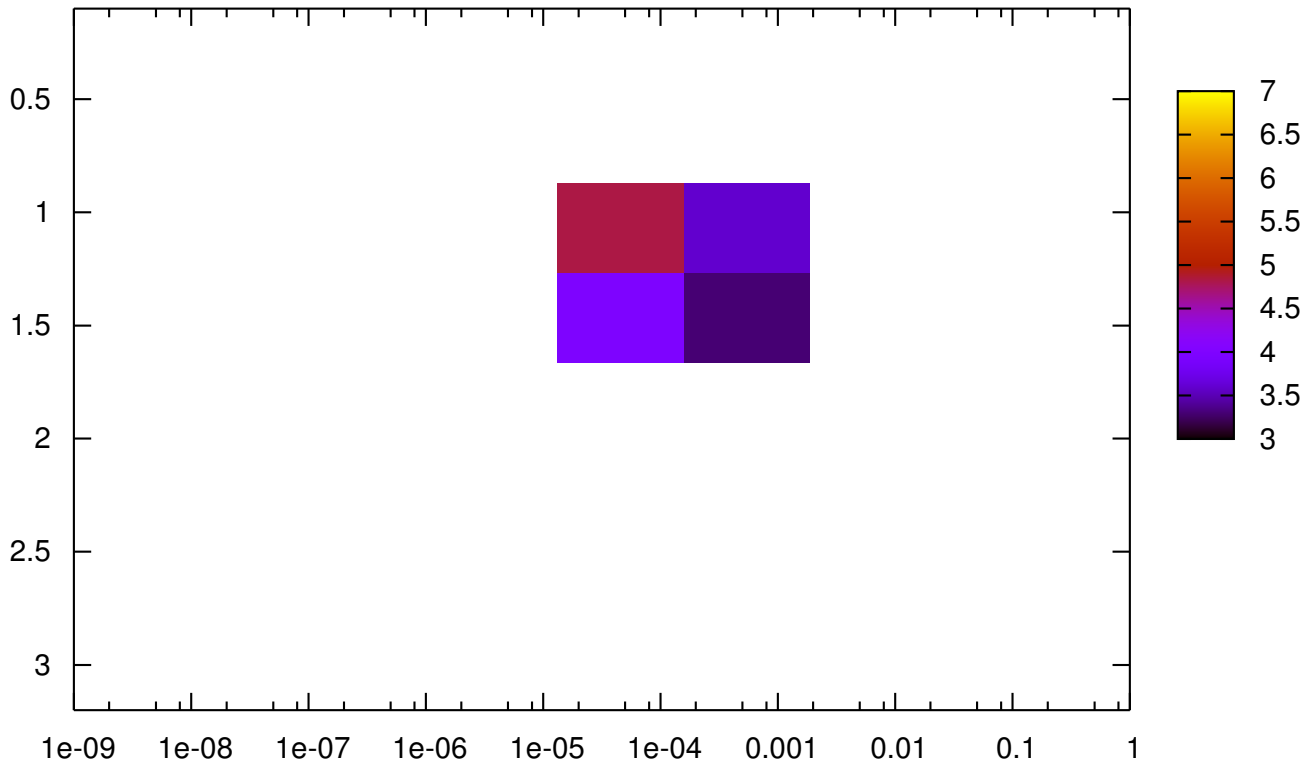
Parameter search - Adaptive Grid



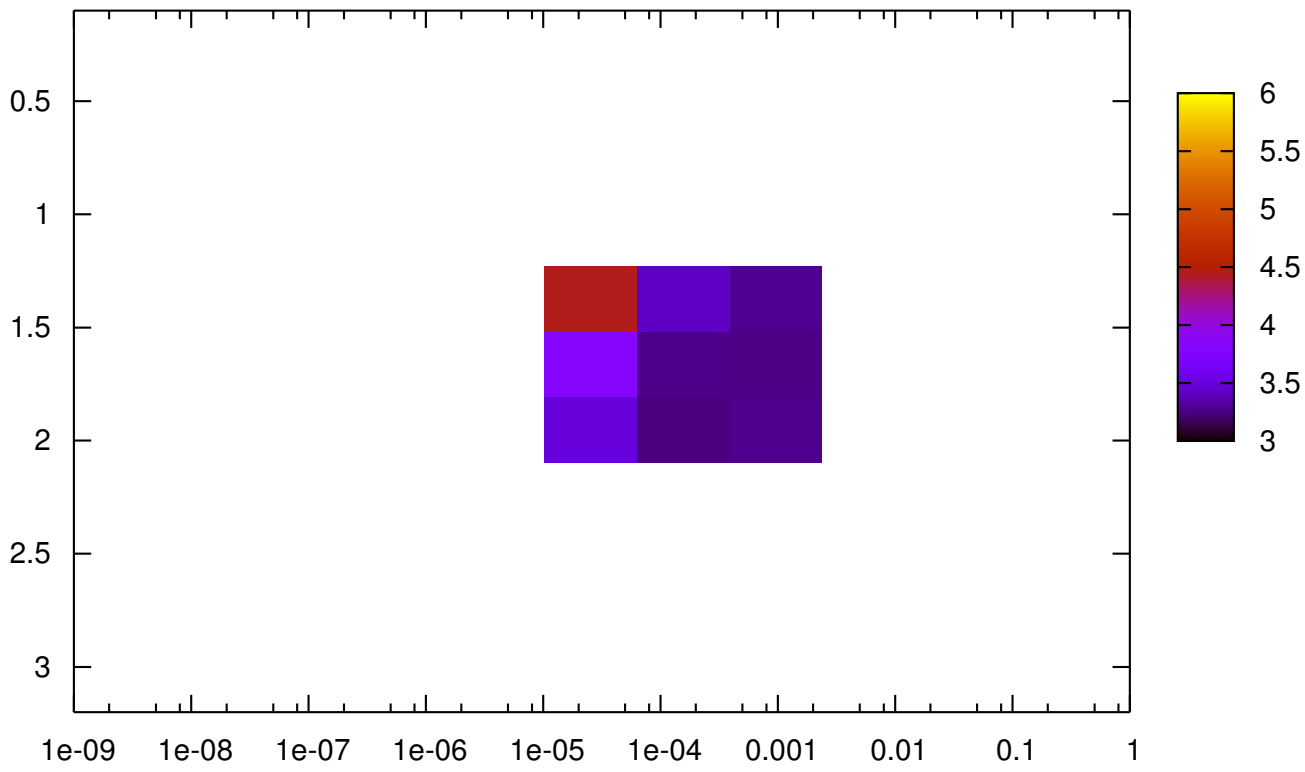
Parameter search - Adaptive Grid



Parameter search - Adaptive Grid



Parameter search - Adaptive Grid



Genetic parameter search

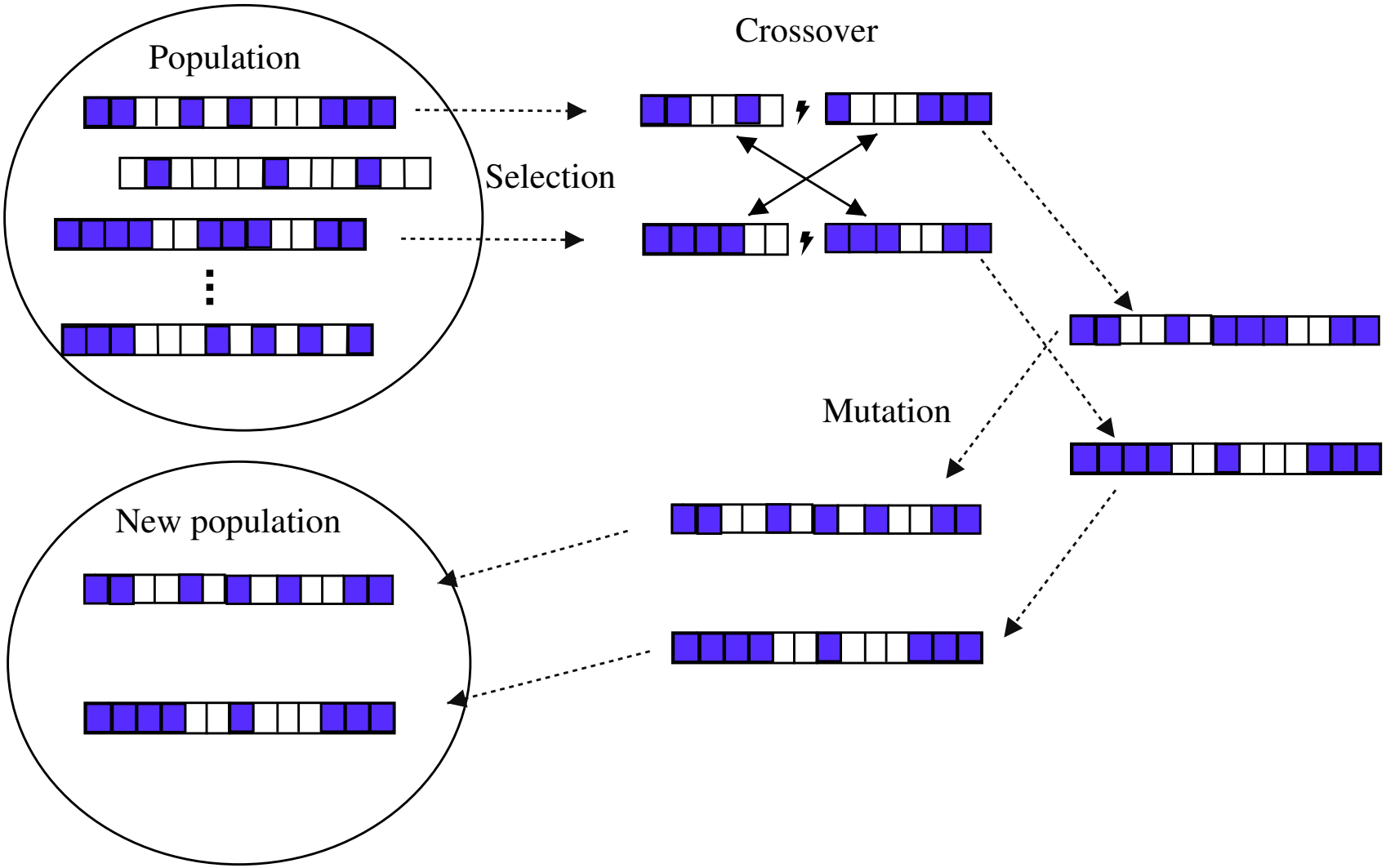
Adaptive grid

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

Genetic algorithms

- applaying simple genetic algorithm
- fitness function - the lower the crossvalidation error the higher the fitness
- random partitioning in crossvalidation
- lazy evaluations

Genetic algorithms (GA)



GA operators for parameter search

Individual

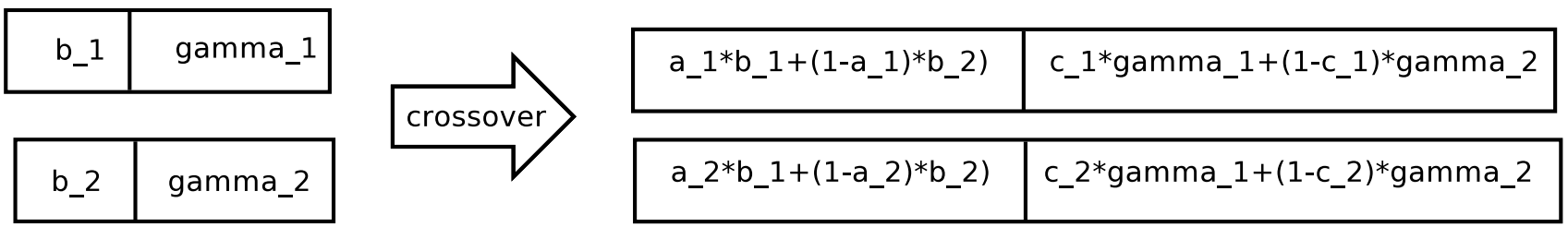
Individual used for search including kernel type:

type of kernel	kernel parameters	reg. parameter

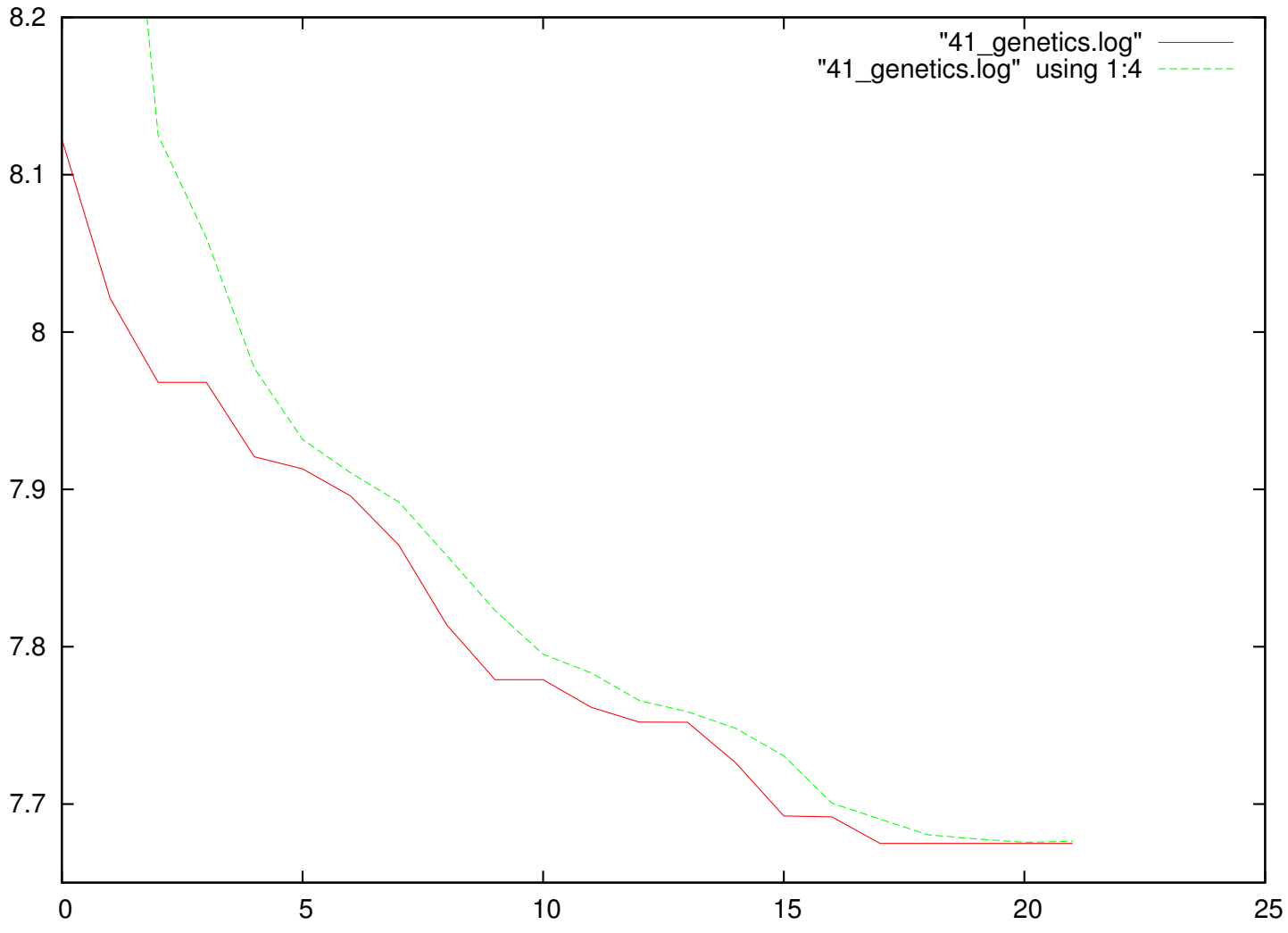
Individual used for Gaussian kernels:

width	reg. parameter

Crossover



Genetic algorithm - example



Experiments

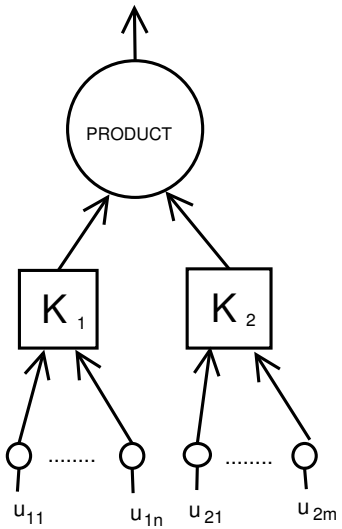
- separate data for training and testing
- find parameters using training data set
- run RN learning using winning parameters
- evaluate error on the testing data set

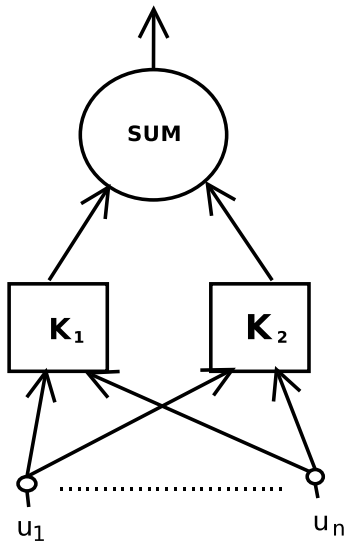
$$E = 100\frac{1}{N}\sum_{i=1}^{N}||\vec{x}^i - f(\vec{x}^i)||^2$$

- **Proben1** data repository
- both approximation and classification tasks
- each task in 3 different partitioning

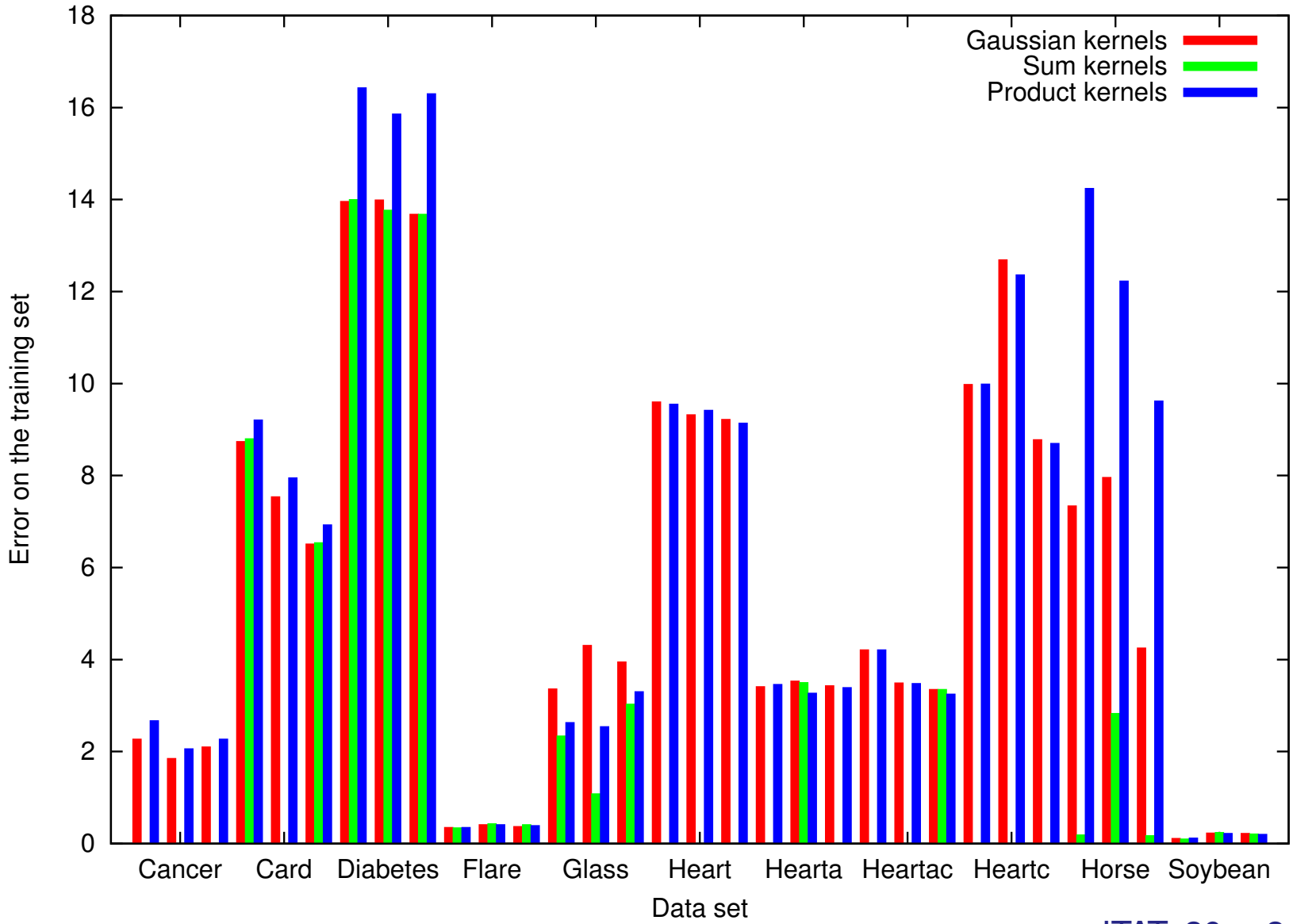
Product and Sum Kernels

- 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. Šámalová)

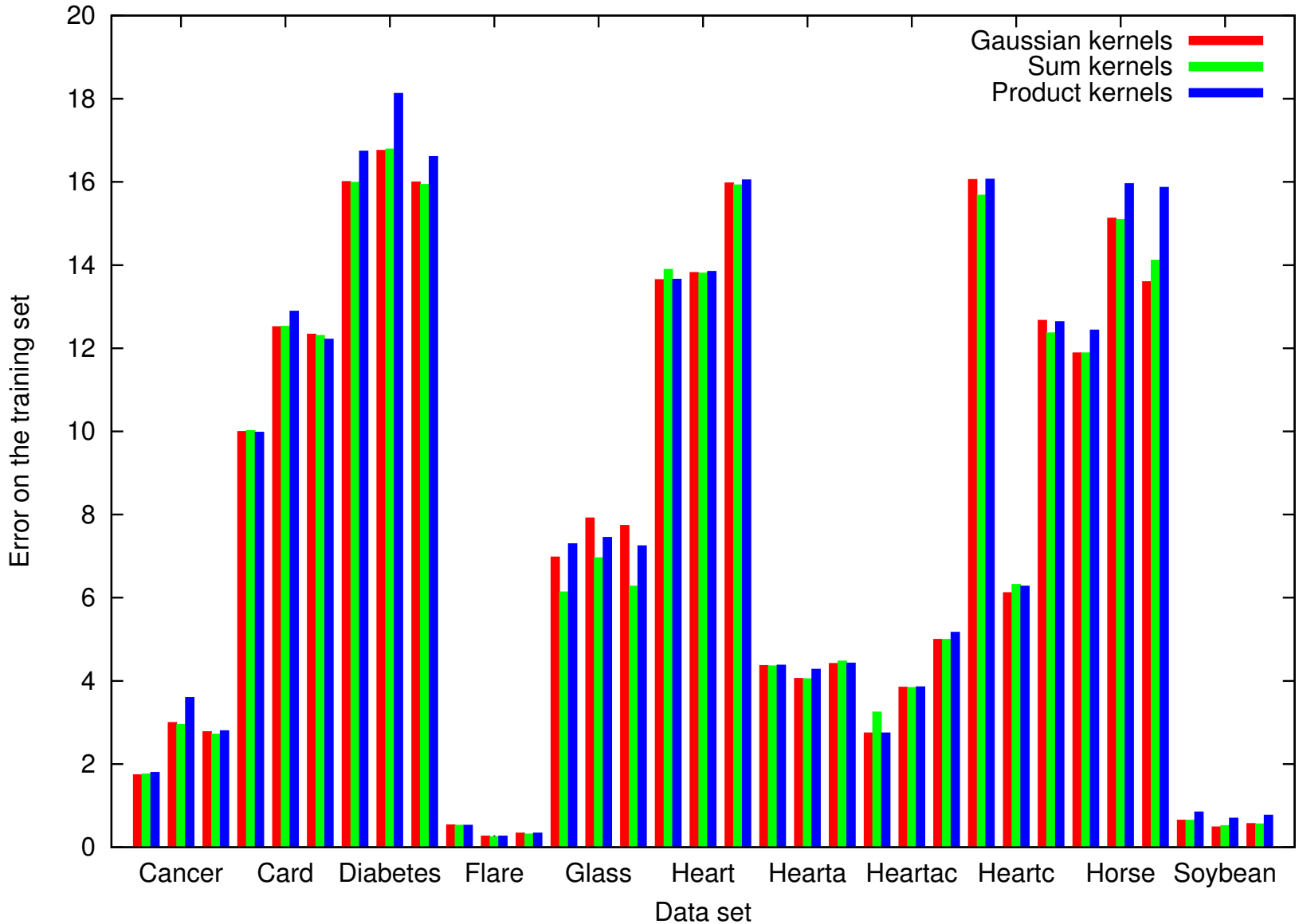




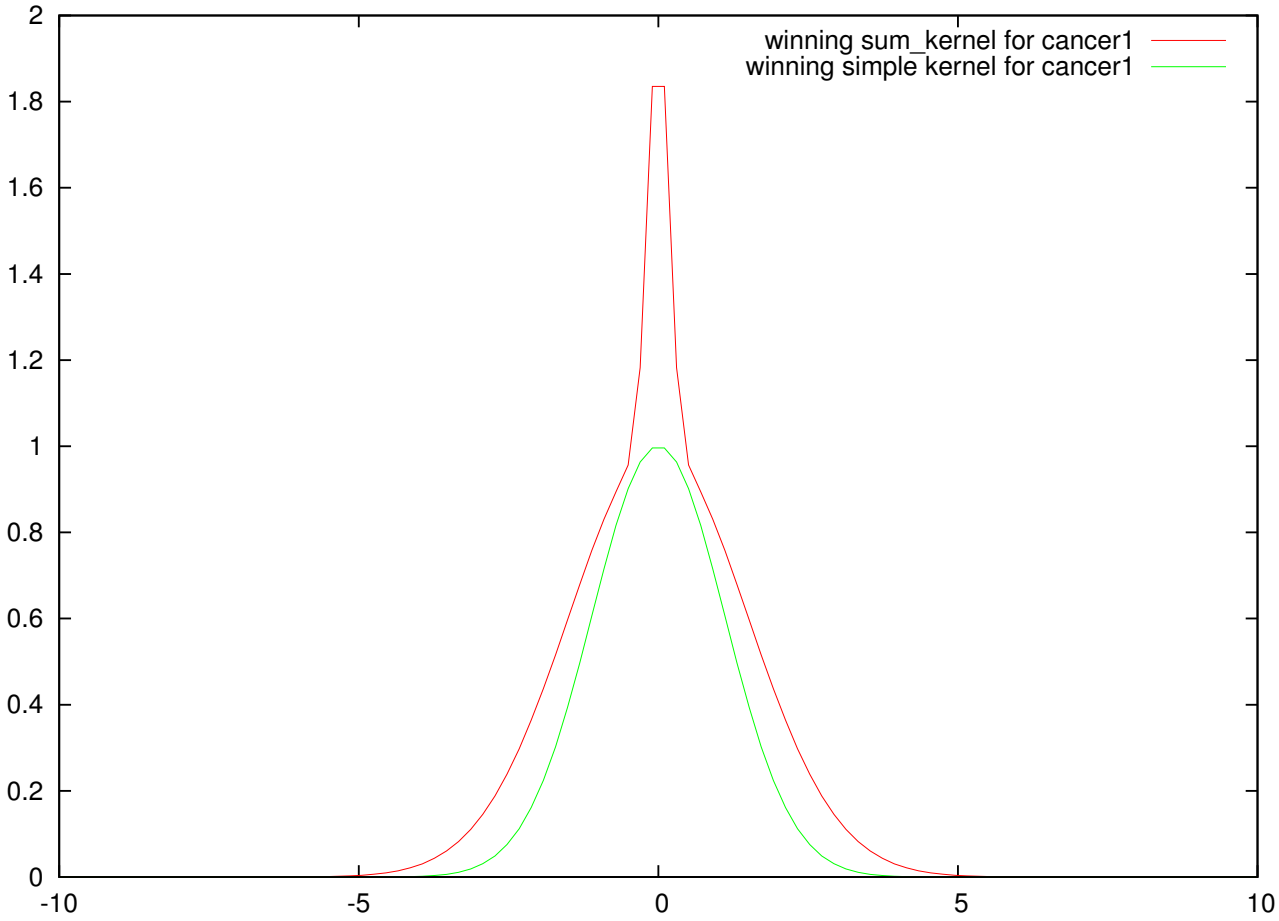
Comparison of Gaussian, Sum and Product kernel



Comparison of Gaussian, Sum and Product kernel



Winning sum unit



Conclusion

Sumary

- RN learning algorithm
- methods for parameter search
- comparison of Gaussian, Product and Sum kernels

Work in progress & future work

- include kernel type in parameter search