Evolution of Composite Kernel Functions for Regularization Networks

Petra Vidnerová

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

May 2011

Outline

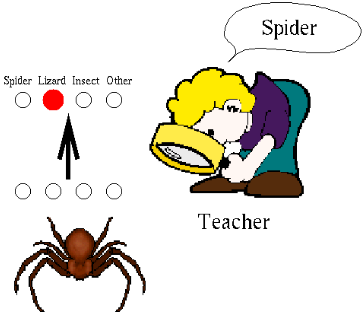
- Introduction - supervised learning
- Regularization networks
- Meta-parameters - kernel function
- Elementary kernels
- Sum and linear combination of kernels
- Product kernels
- Summary and future work

Introduction

Supervised Learning

Learning

- given set of data samples
- find underlying trend, description of data



Supervised learning

- data – input-output patterns
- create model representing IO mapping
- classification, regression, prediction, etc.

Regularization Networks

Regularization Networks

Regularization Networks

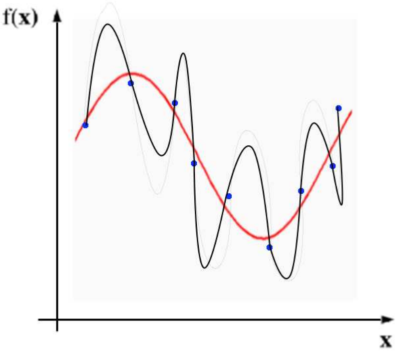
- method for supervised learning
- a family of feedforward neural networks with one hidden layer
- derived from regularization theory
- very good theoretical background

Our Focus

- we are interested in their real applicability
- setup of explicit parameters

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

Regularization approach

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

Derivation of Regularization Network

Stabilizer Based on Fourier Transform

[Girosi, Jones, Poggio, 1995]

- reflects some knowledge about the target function (usually smoothness, etc.)
- penalize functions that oscillate too much
- stabilizer in a form:

$$\Phi[f] = \int_{R^d} d\vec{s} \frac{|\tilde{f}(\vec{s})|^2}{\tilde{G}(\vec{s})}$$

$\tilde{f}$	Fourier transform of f	$\tilde{G}(\vec{s}) \to 0$ for $\|\|s\|\| \to \infty$
$\tilde{G}$	positive function	$1/\tilde{G}$ high-pass filter

Derivation of Regularization Network

Form of the Solution

- for a wide class of stabilizers (G positive semi-definite) the solution has a form

$$f(x) = \sum_{i=1}^{N} w_i G(\vec{x} - \vec{x}_i)$$

- where weights w_i satisfy

$$(\gamma I + G)\vec{w} = \vec{y}$$

- represented by feed-forward neural network with one hidden layer

Derivation of Regularization Network

Using Reproducing Kernel Hilbert Spaces

[Poggio, Smale, 2003]

- 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 $\mathcal{H}_K$

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

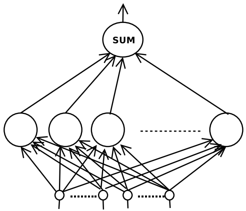
- minimise $H[f]$ over $\mathcal{H}_K$ $\longrightarrow$ solution:

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

Derivation of Regularization Network

Regularization Network

$$f(x) = \sum_{i=1}^{N} w_i G(\vec{x} - \vec{x}_i)$$



- function G called basis or kernel function
- choice of G represents our knowledge or assumption about the problem
- choice of G is crucial for the generalization performance of the network

RN learning algorithm

Basic Algorithm

1. set the centers of kernel functions to the data points
2. compute the output weights by solving linear system

$$(\gamma I + K)\vec{w} = \vec{y}$$

Advantages and Disadvantages

- algorithm simple and effective
- choice of γ and kernel function is crucial for the performance of the algorithm (cross-validation)

Meta-parameters Kernel Function

Meta-parameters

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 usually by user
- kernel parameter and regularization parameter by cross-validation
- in this work: all parameters by genetic approach

Role of Kernel Function

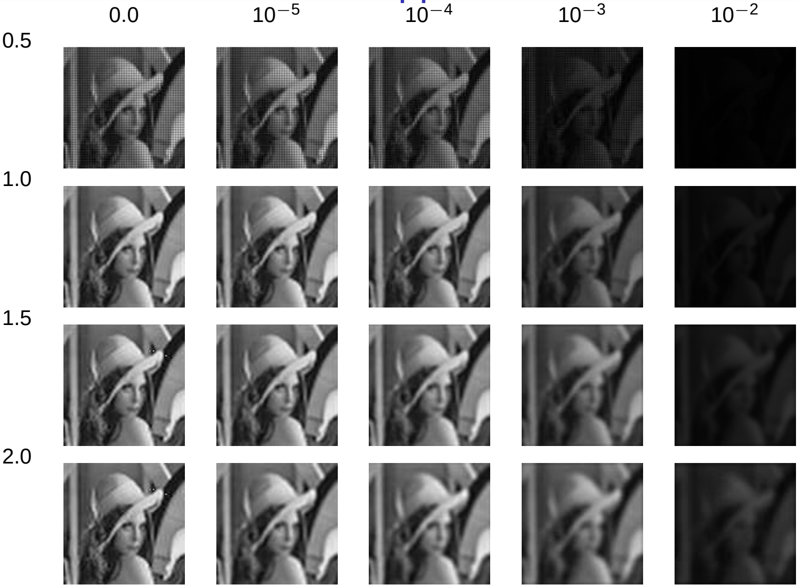
Choice of Kernel Function

- choice of a stabilizer
- choice of a function space for learning (hypothesis space)

Role of Kernel Function

- represent our prior knowledge about the problem
- *no free lunch* in kernel function choice
- should be chosen according to the given problem
- what functions are good first choice?

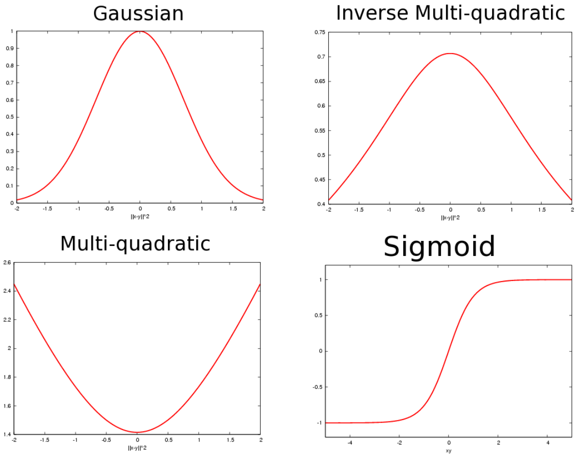
Lenna - approximation



Elementary Kernel Functions

Elementary Kernel Functions

- frequently used kernel functions:



Genetic Parameter Search with Species

Individuals

- individuals coding RN meta-parameters $I = \{K, p, \gamma\}$, i.e. $I = \{\text{Gaussian}, \text{width} = 0.5, \gamma = 0.01\}$.

Individual used for search including kernel type:

type of kernel	kernel parameters	reg. parameter

Individual used for Gaussian kernels:

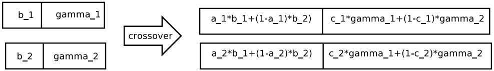
width	reg. parameter

Co-evolution

- subpopulations corresponding to different kernel functions
- selection on the whole population
- crossover on subpopulations

Genetic Parameter Search with Species

Crossover



Mutation

- standard biased mutation

Fitness

- optimize not only precise approximation but also good generalization
- use cross-validation error (10-fold cross-validation)
- the lower cross-validation error the higher fitness

Experiments - Data

- benchmark data sets - Proben1 data repository

Task name	n	m	N_{train}	N_{test}	**Type**
cancer	9	2	525	174	class
card	51	2	518	172	class
diabetes	8	2	576	192	class
flare	24	3	800	266	approx
glass	9	6	161	53	class
heartac	35	1	228	75	approx
hearta	35	1	690	230	approx
heartc	35	2	228	75	class
heart	35	2	690	230	class
horse	58	3	273	91	class

Experiments - Methodology

General rules

- separate data for training and testing
- find suitable kernel and γ on training set by evolution
- learn on training set (estimation of weights w)
- evaluate error on testing set - generalization ability

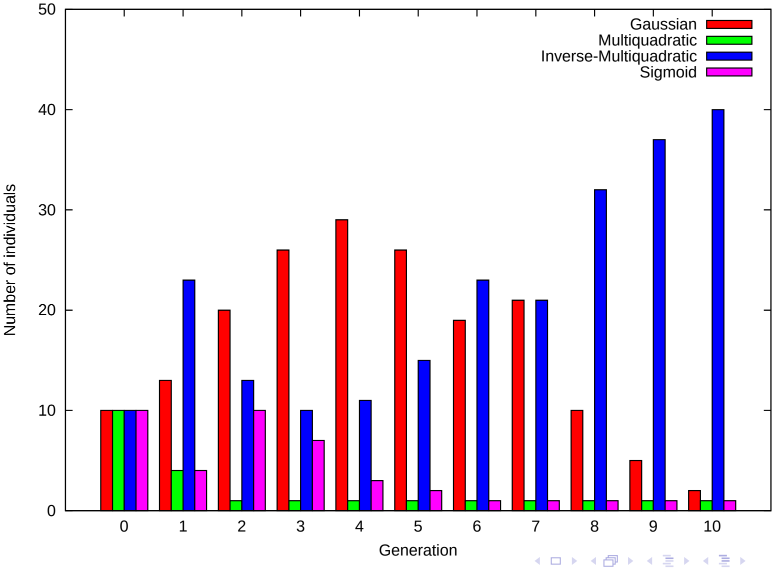
$$E = 100 \frac{1}{Nm} \sum_{i=1}^{N_S} ||\vec{y}_i - f(\vec{x}_i)||^2$$

- evaluate each experiment $10\times$, compare mean values

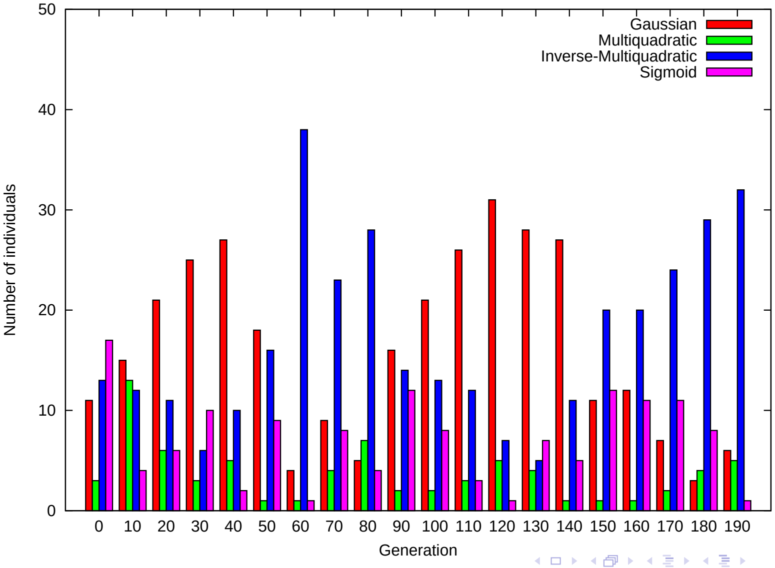
Elementary functions

- initial population - 10 individuals for each kernel
- 200 generations

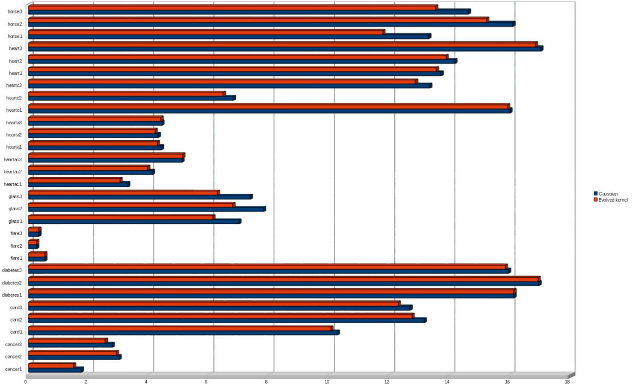
Subpopulations during Evolution - Tournament Selection



Subpopulations during Evolution - Roulette-wheel Selection



Comparison with Gaussian Kernel



- 27 cases inv. multiquadric better than gaussian, 2 cases equal
- on all cases wins inv. multiquadratic, only on diabetes2 two times of 10 gaussian and diabetes3 5 times of 10 gaussian

Sum and Linear Combinations

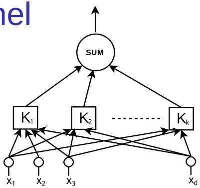
Sum kernel functions

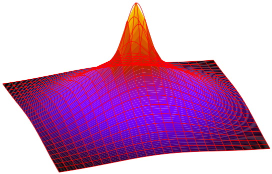
Theory

- based on Aronszajn theory of reproducing kernels
- sum of two RKHS is RKHS
- corresponding kernel function is a sum of the two original kernel functions

$$K(x,y) = K_1(x,y) + K_2(x,y)$$

Sum kernel





Evolution of Sum Kernels

Individuals

$$I = \{ \text{type of kernel function } K_1, \text{kernel parameter}, \text{type of kernel function } K_2, \text{kernel parameter}, \gamma \}$$

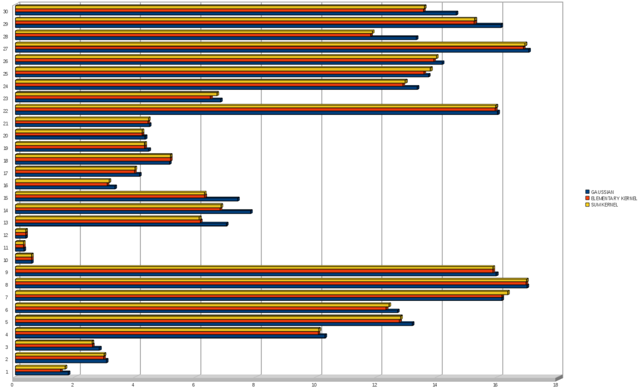
Crossover

- sub-kernels are interchanged

Experiment

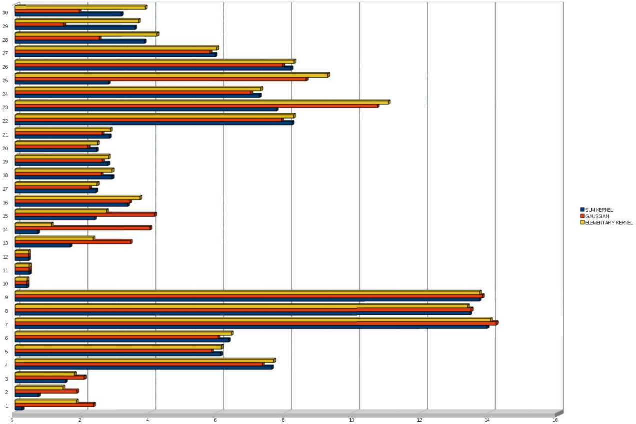
- population of 50 individuals
- 300 generations

Sum Kernels - Test Errors



- sum kernel outperforms gaussian on 27 cases
- sum kernel outperforms inv. multiquadric on 7 cases

Sum Kernels - Training Errors



- sum kernel outperforms gaussian in 13 cases
- sum kernel outperforms inv. multiquadric **in 29 cases**
- on some cases (i.e. *cancer*) very low training errors

Sum Kernels ... are they useful?

Evolved sum kernels

- combination of two inv. multiquadric or inv. multiquadric and gaussian
- one narrow and one wide
- wide kernel function - stress on generalization
- narrow kernel function - precise approximation in training samples

Application

- sum kernels can achieve low training errors without the loss of generalization
- useful for data with low noise

Linear combination of kernels

Linear combination

- generalization of sum kernels
- kernel function is a linear combination of elementary kernels

$$K(x,y) = \alpha K_1(x,y) + \beta K_2(x,y)$$

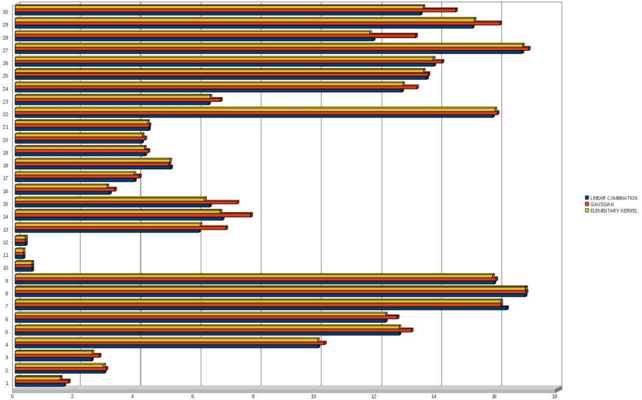
Individuals

$$I = \{ \alpha, \text{type of kernel function } K_1, \text{kernel parameter}, \beta, \text{type of kernel function } K_2, \text{kernel parameter}, \gamma \}$$

Crossover

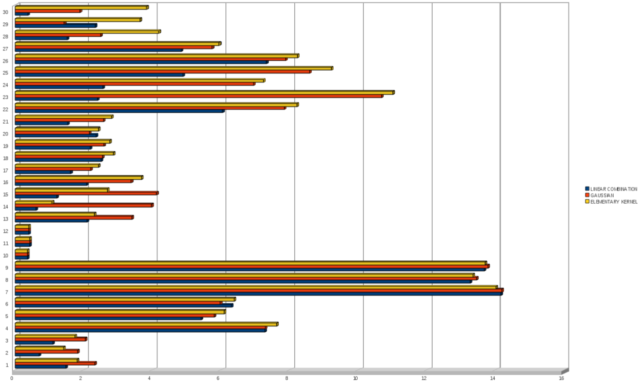
- sub-kernels and coefficients are interchanged

Linear Combinations - Test Errors



- linear combination outperforms gauss in 28 cases
- linear combination outperforms inv. multiquadric in 12 cases

Linear Combinations - Training Errors

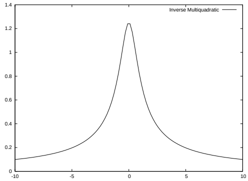


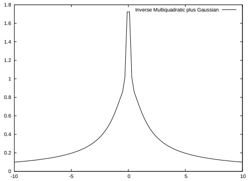
- linear combination outperforms gauss in 24 cases
- linear combination outperforms inv. multiquadric in 28 cases

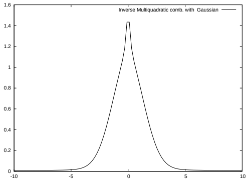
Examples of Sum Kernels and Linear Combinations

Cancer1

Inv. multiquadric		Sum Kernel		Linear Comb.	
E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
1.83	**1.50**	0.01	1.64	0.14	1.53



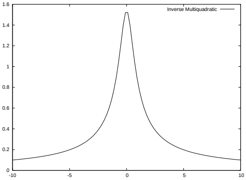


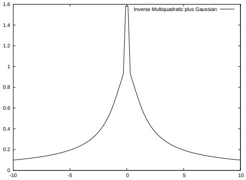


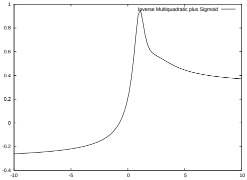
Examples of Sum Kernels and Linear Combinations

Cancer2

Inv. multiquadric		Sum Kernel		Linear Comb.	
E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
1.41	**2.92**	0.01	2.93	1.34	**2.92**



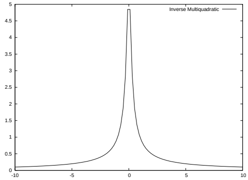


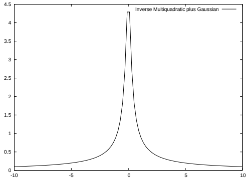


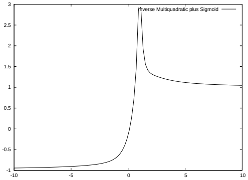
Examples of Sum Kernels and Linear Combinations

Glass1

Inv. multiquadric		Sum Kernel		Linear Comb.	
E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
2.32	6.13	0.02	6.09	2.19	**6.05**







Linear Combination of Kernels - Conclusion

- slightly better results than sum kernels
- similarly to sum kernels combinations mainly of inv. multiquadrics and gaussians
- more parameters to evolve

	Sum Kernel	Composite Kernel
cancer1	Gauss(0.20)+InvMq(1.05)	0.07*InvMq(0.12)+0.99*Gauss(1.98)
cancer2	Gauss(0.15)+InvMq(1.05)	0.55*InvMq(0.49)+0.31*Sgm(1.62)
cancer3	Gauss(1.99)+InvMq(0.72)	0.77*Gauss(0.13)+0.22*Sgm(1.97)
card1	InvMq(1.9)+InvMq(1.99)	0.35*InvMq(1.98)+0.01*Gauss(0.54)
card2	Gauss(1.99)+InvMq(1.79)	0.04*Gauss(0.56)+0.96*InvMq(1.99)
card3	Gauss(1.99)+InvMq(1.99)	0.95*InvMq(1.98)+0.25*InvMq(1.98)
flare1	InvMq(1.99)+InvMq(1.99)	0.19*InvMq(1.97)+0.97*InvMq(1.98)
flare2	Gauss(1.98)+Gauss(1.99)	0.09*InvMq(1.95)+0.72*InvMq(1.98)
flare3	InvMq(1.99)+InvMq(1.99)	0.69*InvMq(1.99)+0.51*InvMq(1.97)
glass1	InvMq(0.21)+Gauss(0.03)	0.51*InvMq(0.16)+0.99*Sgm(0.79)
glass2	Gauss(0.05)+InvMq(0.20)	0.59*Gauss(1.10)+0.11*InvMq(0.11)
glass3	InvMq(0.19)+Sgm(0.44)	0.92*InvMq(0.35)+0.62*Gauss(0.05)
heartac1	InvMq(1.99)+InvMq(1.99)	0.50*InvMq(1.99)+0.05*InvMq(1.96)
heartac2	InvMq(1.99)+Gauss(1.99)	0.22*InvMq(1.96)+0.91*InvMq(1.99)
heartac3	Gauss(1.98)+InvMq(1.99)	0.90*InvMq(1.99)+0.17*Gauss(0.02)
hearta1	Gauss(1.99)+InvMq(1.95)	0.01*Gauss(0.13)+0.65*InvMq(1.98)
hearta2	InvMq(1.99)+Gauss(1.99)	0.02*Sig(0.59)+0.97*InvMq(1.88)
hearta3	InvMq(1.98)+InvMq(1.99)	0.91*InvMq(1.95)+0.07*Gauss(0.05)

Product Kernels

Product Kernel Functions

Theory

- based on Aronszajn theory of reproducing kernels
- product of two RKHS is RKHS
- corresponding kernel function is a product of the two original kernel functions

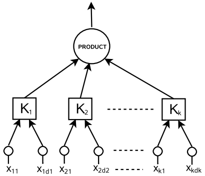
$$K(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k, \vec{y}_1, \vec{y}_2, \ldots, \vec{y}_k) = K_1(\vec{x}_1, \vec{y}_1) K_2(\vec{x}_2, \vec{y}_2) \cdots K_k(\vec{x}_k, \vec{y}_k)$$

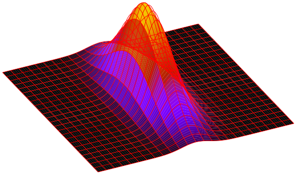
Motivation

- heterogenous data, attributes of different types or qualities
- in product kernel different attributes can be processed by different kernels
- combination of kernels on different data types

Product Kernel

Product Unit





Individuals

$$I = \{ \text{attribute vector } i_1, \ldots, i_n, \\ \text{type of kernel function } K_1, \text{kernel parameter}, \\ \text{type of kernel function } K_2, \text{kernel parameter}, \gamma \},$$

Evolution of Product Kernels

Crossover

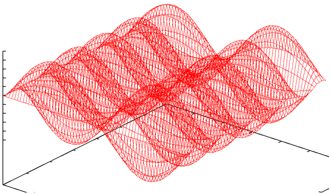
- interchange of sub-kernels
- standard one-point crossover on attribute vectors

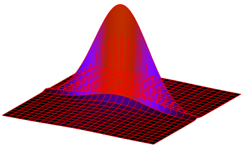
Experiment

- population of 50 individuals
- 300 generations

Simple Example - Approximation of $sin(x)sin(y)$ function

Task

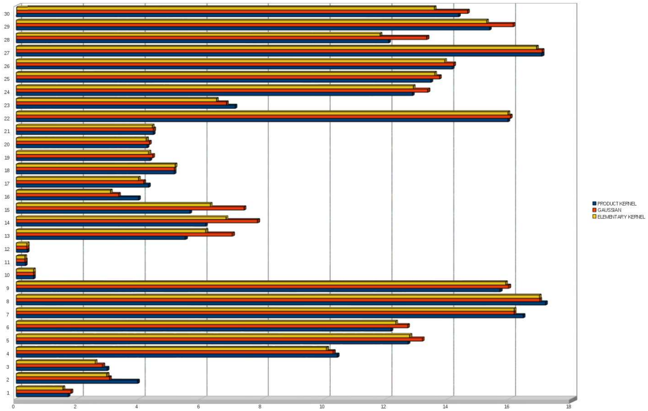




Approximation with Product Kernel

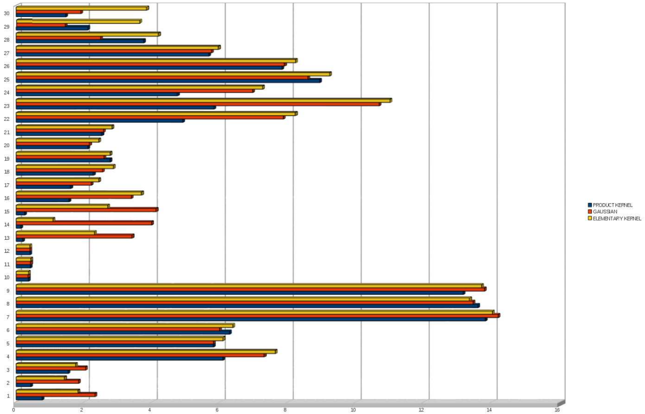
Kernel	E	kernel parameters
Elementary	0.033912	Gauss(p=0.63)
Product	0.000004	Gauss(p=0.50)*Inv_Multiquadric(p=0.02)

Product Kernels - Test Errors



- product outperforms gauss in 19 cases
- product outperforms inv. multiquadric in 10 cases

Product Kernels - Training Errors



- product outperforms gauss in 22 cases
- product outperforms inv. multiquadric in 27 cases

Product Kernels - Conclusion

Evolved Product Kernels

- product of two inv. multiquadrics of different widths or a product of inv. multiquadric and gaussian
- precise approximation of training data
- useful for data with low noise

Applications

- useful for data with low noise
- useful for data with heterogenous attributes
- possible application for data with attributes of different types

Conclusion

Summary and Conclusion

Summary

- learning with RN networks described
- role of kernel function discussed
- composite kernels - sum, linear combination, product

Advantages of composite kernel functions

- accurate approximation while preserving generalization
- combination of narrow and wide kernels suitable for data with low level of noise
- product kernels suitable for data with heterogenous attributes

Possible future work

- kernels on other data types (categorical, strings, etc.)

References

Regularization Networks and Kernel Methods

- Girosi, Jones, and Poggio: *Regularization Theory and Neural Networks Architectures*, 1995, Neural Computation 2/7, p. 219–269.
- Schoelkopf and Smola: *Learning with Kernels*, 2002, MIT Press, Cambridge, Massachusetts.

Evolution of Kernels and Composite Kernels

- P. Vidnerová and R. Neruda: *Genetic Algorithm with Species for Regularization Network Metalearning*, Proceedings of ITAT 2010.
- P. Vidnerová and R. Neruda: *Evolving Sum and Composite Kernel Functions for Regularization Networks*. ICANNGA 2011.
- P. Vidnerová and R. Neruda: *Evolution of Product Kernels for Regularization Networks.* Submitted to ICIC 2011.

Thank you! Questions?