Product Multi-Kernels for Sensor Data Analysis

Petra Vidnerová and Roman Neruda
Institute of Computer Science, Czech Academy of Sciences, Prague

Introduction

Our approach is based on the model called *regularization network* (RN). RNs benefit from a good theoretical background, they has been proved to be the solution of the problem of learning from examples when formulated as a regularized minimization problem [1, 2].

The key step of the RN learning is a choice of kernel function. Different kernel functions are suitable for different data types, but data are often heterogeneous. We proposed the *composite kernel functions* that reflect better the character of data. Such approach can be ranked among the so called *multi-kernel* models.

RN and Kernels

Learning problem formulation

Given the data set $\{(\vec{x}_i, y_i) \in R^n \times R\}_{i=1}^N$, obtained by random sampling of a unknown function f, our goal is to find the function f or its approximation.

Regularization Network

Minimize $H[f] = \frac{1}{N}\sum_{i=1}^N (f(\vec{x}_i) - y_i)^2 + \gamma\Phi[f]$, where Φ is some *stabilizer* and $\gamma > 0$.

The solution is unique and has the form

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

where I is the identity matrix, K is the matrix $K_{i,j} = K(\vec{x_i}, \vec{x_j})$, and $\vec{y} = (y_1, \ldots, y_N)$. Meta-parameters γ, and a type of kernel K are given.

Kernel function

The choice of the kernel function K is an important part of the learning. It corresponds to the choice of a stabilizer and reflects our prior knowledge of the problem.

References

[1] Girosi, F., Jones, M., Poggio, T.: Regularization theory and Neural Networks architectures. Neural Computation **2** (7 1995) 219–269

[2] Kůrková, V.: Some comparisons of networks with radial and kernel units. In: Neural Networks. LNCS 7553, Berlin, Heidelberg, Springer-Verlag (2012) 17–24

[3] Vito, S.D., Massera, E., Piga, M., Martinotto, L., Francia, G.D.: On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sensors and Actuators B: Chemical **129**(2) (2008) 750 – 757

Multi-Kernels

Common elementary kernel functions:

linear	$K(\vec{x}, \vec{y}) = \vec{x}^T\vec{y}$	polynomial	$K(\vec{x}, \vec{y}) = (\alpha\vec{x}^T\vec{y} + r)^d, \alpha > 0$
Gaussian	$K(\vec{x}, \vec{y}) = exp(-\alpha\|\vec{x} - \vec{y}\|^2), \alpha > 0$	sigmoid	$K(\vec{x}, \vec{y}) = \tanh(\alpha\vec{x}^T\vec{y} + r)$

α, d and r are parameters of the kernel.

Motivation for multi-kernel approach stems from the multi-modal nature of data. Each set of features may require a different notion of similarity (i.e., a different kernel). Instead of building a specialized kernel for such applications, it is possible to define just one kernel for each of these modes, and combine them.

It this work, two types of composite kernels are considered:

Product kernels K_1 and K_2 are some kernel functions defined on $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_1}$, $n_1 + n_2 = n$. Then, a product kernel is defined: $K(\vec{x}, \vec{y}) = K(\vec{x_1}, \vec{x_2}, \vec{y_1}, \vec{y_2}) = K_1(\vec{x}_1, \vec{y}_1)K_2(\vec{x}_2, \vec{y}_2)$.

Sum kernels $K(\vec{x}, \vec{y}) = K_1(\vec{x}, \vec{y}) + K_2(\vec{x}, \vec{y})$, where K_1 and K_2 are kernel functions.

We can combine different kernel functions or two kernel functions of the same type but with different parameters, such as two Gaussians of different widths (but the same centre).

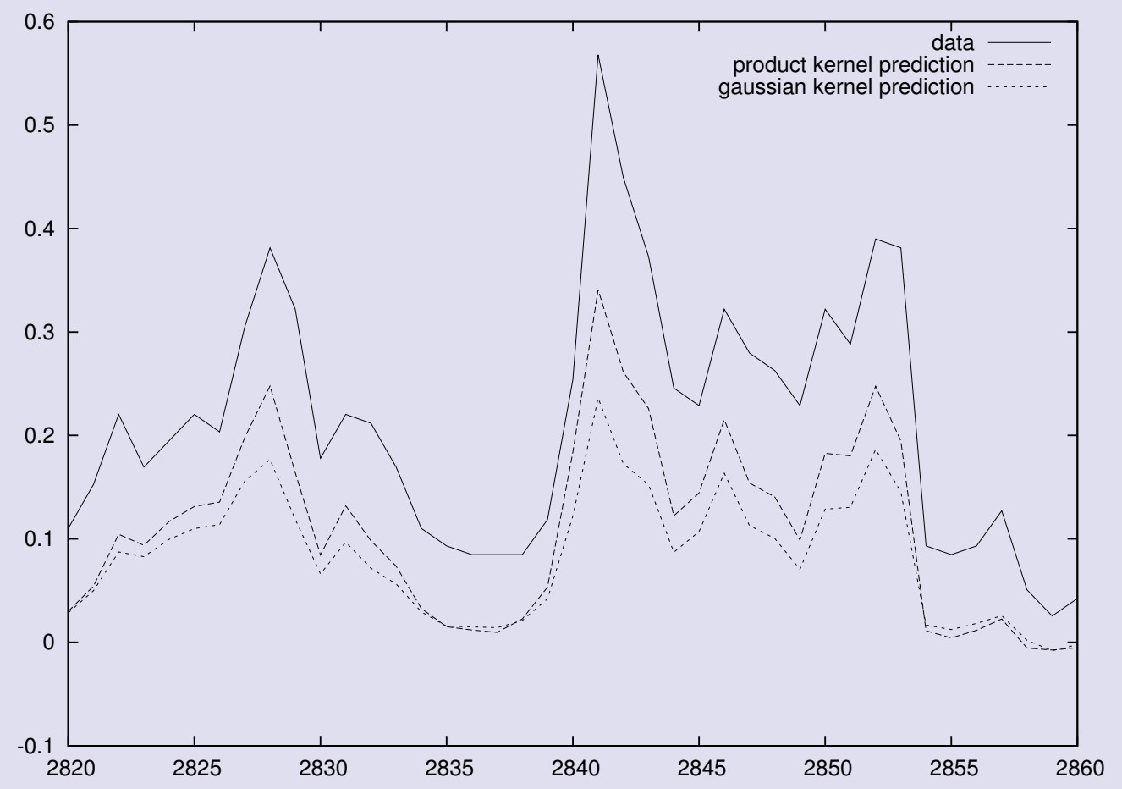
Experiments

Data set: a real-world data from the area of sensor networks for air pollution monitoring [3]. Tens of thousands measurements of gas multi-sensor devices recording concentrations of several gas pollutants for every hour.

5 input sensors and 3 target values(CO, NO_2, and NOx concentrations).

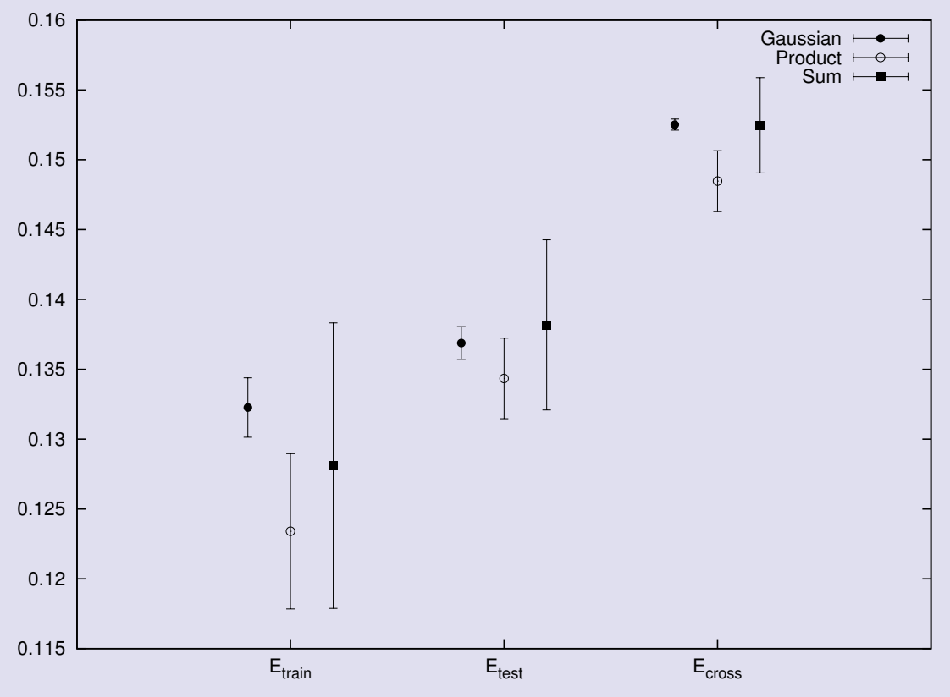
Methodology: GA was run for 300 generations, with 20 individuals, elite size 2. For fitness evaluation, the 10 folds crossvalidation was used.

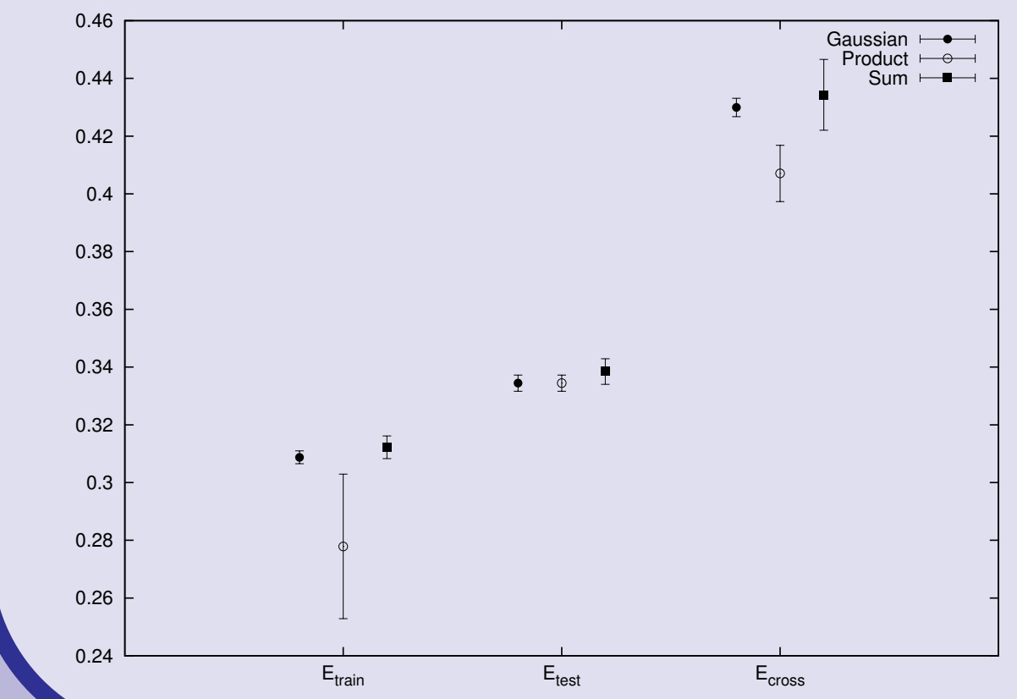
$E = 100\frac{1}{N}\sum_{i=1}^N \|\vec{y}_i - f(\vec{x}_i)\|^2$, each computation was repeated 10 times.

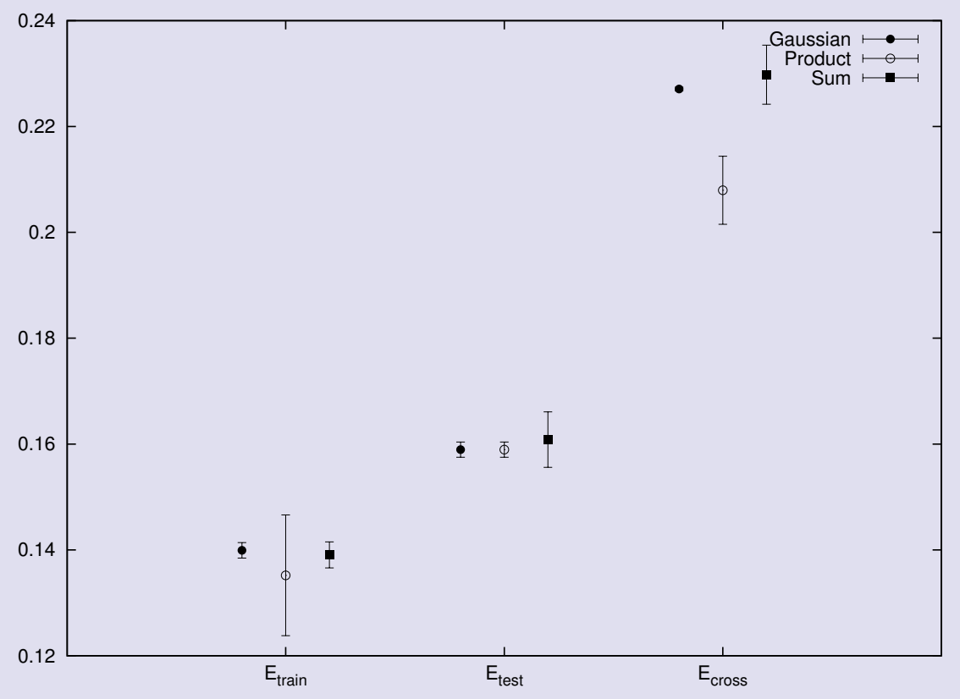


Task I - sparse measurements

The training data consists of 4 samples per day, the rest (values in-between) is then used for testing. Errors for CO (right), NO2 and NOx (bottom left and right).







Acknowledgment

This work was partially supported by the Czech Grant Agency grant 15-18108S and institutional support of the Institute of Computer Science RVO 67985807.

Evolution of Kernels

We deploy genetic algorithm (GA) to search for optimal composite kernels. It works with population of possible kernels (individuals) and evolves them using operators *selection*, *crossover* and *mutation*.

Individual Encoding

Elementary kernel function: $I = \{K, p, \gamma\}$,.

Product kernel: $I = \{K_0, p_0, K_1, p_1, i_1, \ldots, i_n, \gamma\}$,.

Example:

$$\begin{aligned} I = \{&\texttt{Gaussian}, 0.84, \\ &\texttt{Inverse\_Multiquadric}, 1.58, \\ &[0, 0, 1, 0, 1, 1, 1, 1], \gamma = 0.2\}. \end{aligned}$$

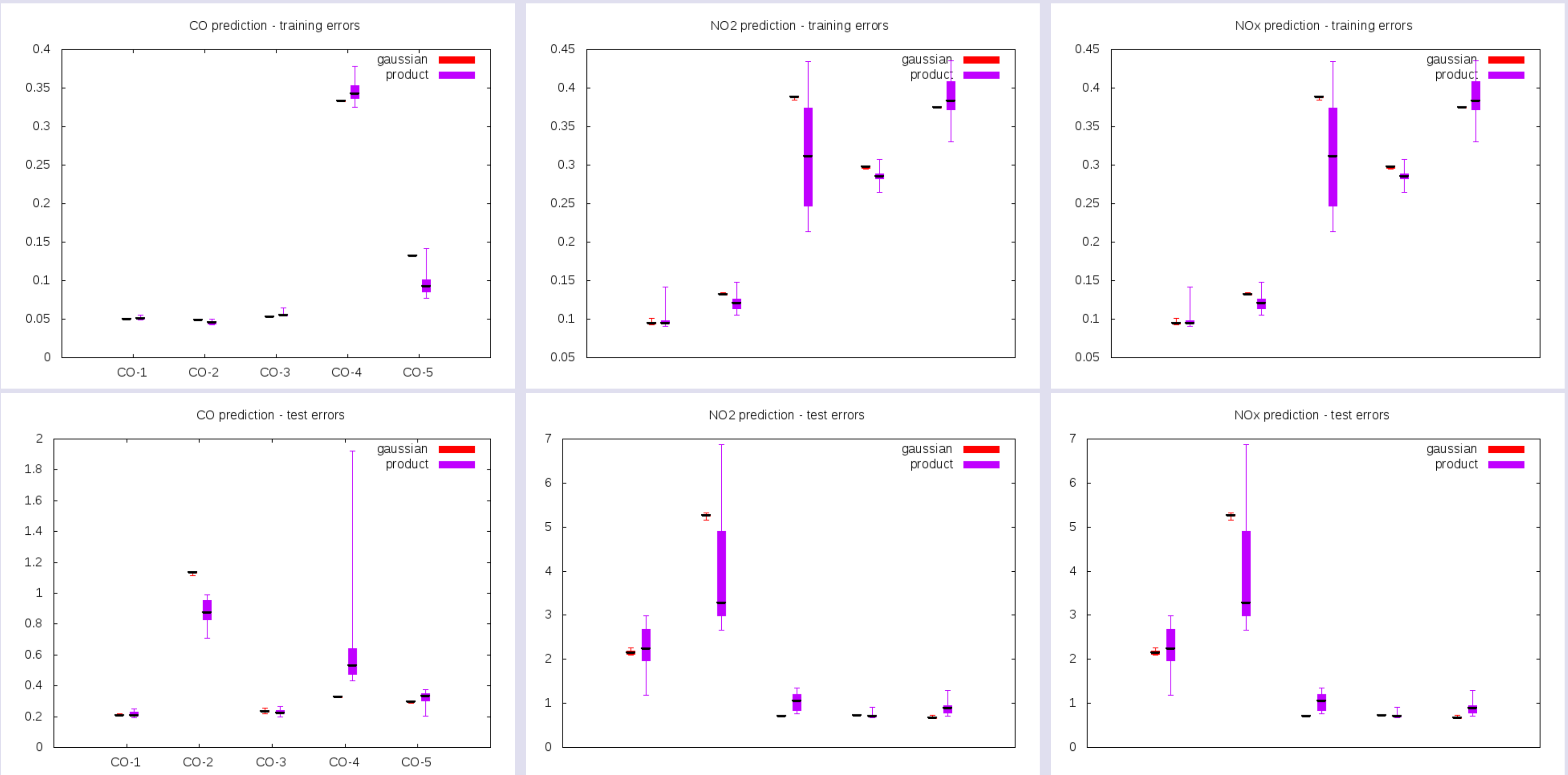
Crossover and Mutation Crossover on elementary kernels generates new values in the interval formed by the parents, i.e. $\gamma = (1 - r)\gamma_1 + r\gamma_2$, where $r \in \langle 0, 1\rangle$ is a random number, γ_1 and γ_2 are parents' values.

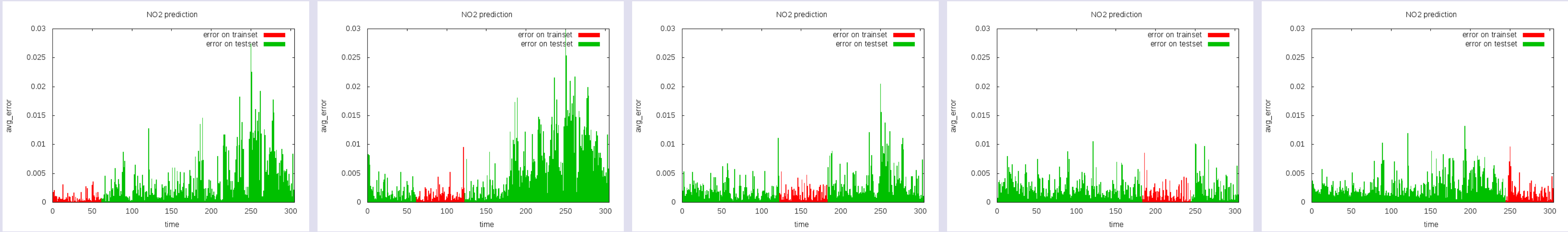
Crossover on composite kernels swaps the subkernels (and in case of product kernels runs one-point crossover on attribute vectors).

Tournament Selection Use the kernel to create a network. Compute the *crossvalidation error*. The winner is the one with the lower error.

Task II - missing epochs

The whole time period was split into five intervals, one for training, the rest for testing. Considering mean values, improvement was achieved mainly on train errors. Minimal errors of product kernels are more promising, the GA should be improved. The resulting product kernels are mainly combinations of Gaussians and Inverse-Multiquadrics.


CO prediction - training errors
NO2 prediction - training errors
NOx prediction - training errors
CO prediction - test errors
NO2 prediction - test errors
NOx prediction - test errors
gaussian 🟥
product 🟪
CO-1
CO-2
CO-3
CO-4
CO-5


NO2 prediction
error on trainset 🟥
error on testset 🟩
avg_error
time