Adaptive metric kernel regression

Cyril Goutte*, Jan Larsen
Department of Mathematical Modelling
Technical University of Denmark
2800 Lyngby, Denmark
cg,jl@imm.dtu.dk

September 10, 1999

Abstract

Kernel smoothing is a widely used non-parametric pattern recognition technique. By nature, it suffers from the curse of dimensionality and is usually difficult to apply to high input dimensions. In this contribution, we propose an algorithm that adapts the input metric used in multivariate regression by minimising a cross-validation estimate of the generalisation error. This allows to automatically adjust the importance of different dimensions. The improvement in terms of modelling performance is illustrated on a variable selection task where the adaptive metric kernel clearly outperforms the standard approach. Finally, we benchmark the method using the DELVE environment.

Keywords: Local regression, kernel smoothing, adaptive metric, cross-validation.

1 Introduction

Neural Networks are often referred to as a non-parametric model. Their popularity is probably in part linked to the fact that there are many cases in which

*Corresponding author

one tries to model some input-output relationship on the basis of empirical data, without any parametric model of the underlying phenomenon. Kernel methods are the archetypal non-parametric method [1], a well-known tool for eg pattern recognition [2]. They have been "re-invented" in the neural networks literature on several occasions [3, 4], for density estimation, classification and regression purposes. However, it has been consistently noted that they suffer badly from the "curse-of-dimensionality" [5], ie produce poor estimators when the input dimension increases.

In this contribution, we will address a possible improvement on the traditional multivariate kernel method. We will mainly be concerned with regression estimation, but the method presented below applies to classification tasks in a straightforward manner. We will first present some general result on the uni- and multivariate kernel regression estimation. We then introduce our method, based on the adaptive estimation of the feature space metric used by the kernels. We perform a number of experiments on a regression task where a number of irrelevant dimensions are added to the input space. The effect of the adaptive metric approach is illustrated on a toy dataset with irrelevant inputs by the fact that its performance on the full input space is better than a standard kernel method using only the relevant input information. Finally, the benchmark of the method using the DELVE environment shows that it performs quite well compared to neural networks and Gaussian Processes.

2 Kernel regression

Let us consider a standard regression problem: from a number N of input-output pairs $\left(\mathbf{x}^{(k)}, y^{(k)}\right)$ sampled from an unknown input-output joint distribution, we wish to learn the relation which, for a new input $\mathbf{x}$, maps an estimated output $\widehat{y}$. Kernel regression is a widely used non-parametric regression method [1]. Probably the best-known univariate kernel smoother derives from the Parzen window density estimator. It takes the form of the Nadaraya-Watson estimator [6, 7], which estimates the expected value $\mathrm{E}(y|x)$ of the output y given a

one-dimensional input x:

$$\widehat{y} = f_\sigma(x) = \frac{\sum_{k=1}^{N} K_\sigma\left(x - x^{(k)}\right) y^{(k)}}{\sum_{k=1}^{N} K_\sigma\left(x - x^{(k)}\right)} \tag{1}$$

where $K_\sigma(x) = K(x/\sigma)/\sigma$ is the scaled kernel with *bandwidth* σ. $K(x)$ is the kernel shape, usually a continuous, bounded and integrable function. Figure 1 displays several kernel shapes which have been scaled to integrate to 1 for easy visual comparison. The actual scaling of the kernel is irrelevant when evaluating equation (1) as the scaling terms in the denominator and the numerator cancel out. Equation 1 is also obtained as the solution of the locally constant regression, weighted by the kernel weights $K_\sigma\left(x - x^{(k)}\right)$. The Nadaraya-Watson estimator has been rediscovered several times in the Neural Networks literature, most recently in [3], and usually in limited forms (eg Gaussian kernel shapes only). Kernels with compact supports are uncommon in a neural networks context, though they are usually considered to have an edge in terms of computational cost [1].

The extension of (1) to the multivariate case is straightforward with the definition of a multivariate kernel. We will here consider only the *spherically symmetric* kernel. For a P dimensional input vector $\mathbf{x}$ and a $P \times P$ symmetric, positive definite *bandwidth matrix* $\mathbf{\Sigma}$, the spherically symmetric kernel is defined from the univariate kernel K as:

$$K_{\mathbf{\Sigma}}(\mathbf{x}) \propto \frac{1}{\sqrt{|\mathbf{\Sigma}|}} K\left(\sqrt{\mathbf{x}^\top \mathbf{\Sigma}^{-1} \mathbf{x}}\right) \tag{2}$$

The analysis of the convergence of the multivariate kernel estimator for general matrices $\mathbf{\Sigma}$ is complicated. For the simpler case where $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$, $\sigma^2 \neq 0$, it is possible to show that the asymptotic rate of convergence of the average generalisation error is $N^{-4/(P+4)}$, where N is the sample size, and assuming additive noise (eg [8] section 4.3, 5.9). As a comparison, the rate of convergence for parametric methods will typically be $N^{-s/P}$ (eg [9] section 4.3), where s measures the smoothness of the underlying function. These theoretical results illustrate the *curse-of-dimensionality*: for high dimensional inputs (large P), the rate of convergence $N^{-4/(P+4)}$ becomes very slow. Furthermore, according to these convergence rates, for reasonably smooth function, the parametric

approach will converge faster than the non-parametric estimator.

A number of methods have been devised to automatically set the bandwidth parameter in a univariate setting (eg [8] chapter 3). The multivariate case is usually considered an extension of the univariate case. Likewise, a typical extension is to use a single bandwidth parameter for multivariate regression. We will now present a scheme that is specific to the multivariate nature of the regression.

3 Adaptive metric

Let us now rewrite equation (1) for multivariate inputs using the squared distance $d^2_{\mathbf{H}}(\mathbf{u}, \mathbf{v}) = (\mathbf{u} - \mathbf{v})^\top \mathbf{H}(\mathbf{u} - \mathbf{v})$, parameterised by the positive definite, symmetric, square matrix $\mathbf{H}$, yielding:

$$\widehat{y} = f_{\mathbf{H}}(\mathbf{x}) = \frac{\sum_{k=1}^{N} K\left(d_{\mathbf{H}}\left(\mathbf{x}, \mathbf{x}^{(k)}\right)\right) y^{(k)}}{\sum_{k=1}^{N} K\left(d_{\mathbf{H}}\left(\mathbf{x}, \mathbf{x}^{(k)}\right)\right)} \tag{3}$$

$\mathbf{H}$ defines the metric in the input space, and is equivalent to the inverse bandwidth matrix in equation (2), $\mathbf{H} \equiv \mathbf{\Sigma}^{-1}$. We will here limit ourselves to a diagonal matrix $\mathbf{H}$ with positive elements parameterised as follows:

$$\mathbf{H} = \operatorname{diag}\left(h_1^2, h_2^2, \ldots, h_P^2\right) = \begin{bmatrix} h_1^2 & 0 & \ldots & 0 \\ 0 & h_2^2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & h_P^2 \end{bmatrix} \tag{4}$$

The diagonal matrix introduces more flexibility than the choice of a single-parameter spherical bandwidth $\mathbf{H} = h^2\mathbf{I}$, but clearly it is less flexible than a full matrix. The parameterisation of a general positive definite matrix (eg by Cholesky decomposition) is more complex, but the derivations presented below can be extended in a straightforward manner. A full matrix will allow to model, among other thing, correlations between the inputs. However, it should be noted that a full parameterisation requires a number of parameters that grows like the square of the input dimension, thus rapidly becoming impractical for limited data. An intermediate solution is to allow a limited amount of correlation

between the input, in which case the challenge lies in finding a parameterisation that ensures that the matrix stays positive definite.

For a diagonal metric, equation (4) ensures that the diagonal elements stay positive for all h_p, hence a positive matrix $\mathbf{H}$. Other parameterisation are possible, among others as $\exp(h)$. Note that the h_p play a role that is inverse to that of the bandwidth in (1). A small h_p corresponds to an irrelevant dimension, ie a broad kernel, while a large h_p reflects an important contribution from the corresponding dimension, akin to a small bandwidth in the traditional approach. In the following, h_p will be referred to as *smoothing* parameter, and h_p^2 as *metric* parameter.

Among the many possible methods for automatically setting the amount of smoothing, we will focus on a traditional approach in statistical learning, namely minimising an estimator of the (average) generalisation error (eg [10], chapter 3). The rationale for this choice is two-fold: first it is easier to relate to the training of other models like linear models or neural networks, than with eg plug-in estimators of the bandwidth [8]; second and most importantly, we derive the gradient of the generalisation estimator wrt the metric parameters which, in conjunction with first-order multi-dimensional minimisation, leads to an original method for setting the bandwidth.

The V-fold cross-validation (XV) estimator [11] is calculated by randomly splitting the N data into V disjoint subsets $S_1, \dots S_V$ of approximately equal sizes, such that $\bigcup_{i=1}^{V} S_i = \{1, \dots N\}$, and averaging the error observed on one set for the estimator obtained on the rest on the data:

$$\widehat{G}_{XV} = \frac{1}{N} \sum_{j=1}^{V} \left[\sum_{i \in S_j} \left(y^{(i)} - f_{\mathbf{H}}^{(j)} \left(\mathbf{x}^{(i)} \right) \right)^2 \right] \tag{5}$$

where $f_{\mathbf{H}}^{(j)}$ is the kernel estimator excluding set S_j:

$$f_{\mathbf{H}}^{(j)} \left(\mathbf{x} \right) = \frac{\sum_{k \notin S_j} K \left(d_{\mathbf{H}} \left(\mathbf{x}, \mathbf{x}^{(k)} \right) \right) y^{(k)}}{\sum_{k \notin S_j} K \left(d_{\mathbf{H}} \left(\mathbf{x}, \mathbf{x}^{(k)} \right) \right)}$$

Equation (5) uses a squared loss function, a common choice that is consistent with the estimation of the expected value of the output y given $\mathbf{x}$. The use of other losses is straightforward. The derivatives of the cross-validation estimator

$\widehat{G}_{XV}$ w.r.t. the smoothing parameters h_p turn out to have an exact analytical expression:

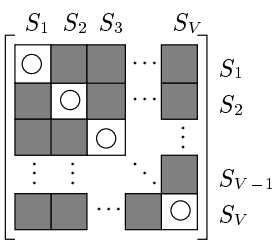
$$\frac{\partial \widehat{G}_{XV}}{\partial h_p} = -\frac{2}{N} \sum_{j=1}^{V} \sum_{i \in S_j} \left(y^{(i)} - f_{\mathbf{H}}^{(j)} \left(\mathbf{x}^{(i)} \right) \right) \frac{\sum_{k \notin S_j} \left(y^{(k)} - f_{\mathbf{H}}^{(j)} \left(\mathbf{x}^{(i)} \right) \right) \frac{\partial K_{ik}}{\partial h_p}}{\sum_{k \notin S_j} K_{ik}} \quad (6)$$

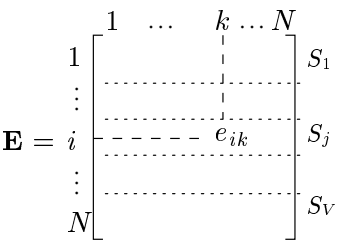
where $K_{ik} = K\left(d_{\mathbf{H}}\left(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}\right)\right)$. In the case of neural networks, a similar expression, namely that of the cross-validation derivatives w.r.t. regularisation parameters, can also be derived [12]. Equation (6) depends on the derivative of the kernel weights $K\left(d_{\mathbf{H}}\left(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}\right)\right)$ w.r.t. the smoothing coefficients h_p. This derivative depends on the kernel shapes, and the expressions for some common kernel shapes are reported in table 1. Note that the h_p that factors in front of each derivative is actually the (half) derivative of the metric parameter h_p^2 w.r.t. to the smoothing parameter h_p. We will adapt the expressions in table 1 for other types of parameterisation by replacing $2h_p$ with the corresponding derivative, eg $\exp(h_p)$ if the metric diagonal is parameterised with exponentials.

The tuning of the smoothing parameters now becomes a first-order multi-dimensional optimisation problem, where we are trying to minimise $\widehat{G}_{XV}$ in the P dimensional metric space. This can be performed by a number of methods such as conjugate gradient or quasi-Newton methods [13].

Implementation.

The straightforward application of (5) and (6) seems complex due to the many pair distances to consider. However, the calculation of the cross-validation can be eased by considering the $N \times N$ matrix $\mathbf{K}$ with elements K_{ik}. For consistency, we define $K_{ik} = 0$ if i and k belong to the same subset S_j. We also define the $N \times N$ matrix $\mathbf{E} = \left[e_{ik} = \left(y^{(k)} - f_{\mathbf{H}}^{(j)} \left(\mathbf{x}^{(i)} \right) \right) \right]$ of cross-validation errors:

$\mathbf{K} =$ 

$\mathbf{E} =$ 

Equation (5) becomes:

$$\widehat{G}_{XV} = \frac{1}{N} \left\| \mathbf{Y} - \operatorname{diag}(\mathbf{K} \cdot \mathbf{1})^{-1} \mathbf{K}\mathbf{Y} \right\|^2$$

where $\mathbf{Y}$ is the matrix of outputs (a column vector for one dimensional outputs) and $\mathbf{1}$ is a column vector of ones. Note that the operation above actually performs an element-by-element division of two vectors such that there is actually no matrix inversion involved. The cross-validation estimator is therefore calculated in $\mathcal{O}(N^2)$ time. Note also that when N is large, $\mathbf{K}$ can be split in horizontal slabs to fit in memory. The gradient of $\widehat{G}_{XV}$ is also expressed in a simple matrix format:

$$\frac{\partial \widehat{G}_{XV}}{\partial h_p} = -\frac{2}{N} \operatorname{diag}(\mathbf{E})^\top \operatorname{diag}\left(\operatorname{diag}(\mathbf{K} \cdot \mathbf{1})^{-1} \mathbf{E} \frac{\partial \mathbf{K}}{\partial h_p} \right) \tag{7}$$

The diagonal operator $\operatorname{diag}(\cdot)$ returns either a vector of diagonal elements when the input is a square matrix, or a square diagonal matrix when the input is a vector. As we consider only diagonal elements again, equation (7) is calculated in $\mathcal{O}(N^2)$ time. The full calculation of the cross-validation error and its derivatives is therefore $\mathcal{O}(PN^2)$. For V-fold cross-validation, only $\frac{V-1}{V}N$ data samples are available for estimating the regressions $\widehat{y}$. In parametric models, this is justified by the fact that performing successively V regressions can be computationally costly, hence the need for V to remain relatively low. In a non-parametric setting such as kernel regression, this is not an issue, as the estimation is $\mathcal{O}(PN^2)$ whatever the value of V. Finally, note that the smoothing parameter depends on the amount of data. If V-fold cross-validation provides an estimate h_V of a smoothing parameter, the value h used for estimating the regression on the basis of the entire dataset will usually be underestimated. This can be corrected using an asymptotic argument based on the volume encompassed by the corresponding kernel. For V-fold cross-validation, the number of data encompassed in the equivalent kernel is proportional to $\frac{V-1}{V} N h_V^{-1/P}$, while on the full dataset, it is $Nh^{-1/P}$. Requiring that both models average roughly over the same amount of data suggests to correct the overall value as $h \equiv (1 - 1/V)^{-1/P} h_V$. When V tends to N, $(1 - 1/V)^{-1/P} \approx 1 + 1/(NP)$ for large N, which approaches 1 and is thus irrelevant.

4 Illustration

Let us now perform some input selection experiments, in which we illustrate the effect of the adaptive metric. We use a dataset generated by a system described in [14]:

$$y = 10 \sin(\pi x_1 x_2) + 20 \left(x_3 - \frac{1}{2} \right)^2 + 10 x_4 + 5 x_5 + \epsilon \tag{8}$$

with Gaussian noise $\epsilon \sim \mathcal{N}(0,1)$. The input vector contains 10 uniformly distributed values, $x_1 \dots x_{10}$, $x_i \sim \mathcal{U}([0,1])$. It is obvious from equation (8) that the last five inputs are irrelevant. However, the input space has 10 dimensions, which can be considered rather large for classical kernel regression. In order to test several small sample conditions we will consider 4 training set sizes: 50, 100, 200 and 500 samples. In order to test the generalisation abilities, we generate a large test set containing 5000 samples. It should be noted that 500 data points is still a small sample for a non-parametric estimation of this system, even if we limit ourselves to the relevant 5-dimensional space. If we wanted to estimate this function using a mesh of regularly spaced points on each dimension, we would need more than 3000 points to map the uniformly sampled interval [0;1] with 5 points on each of 5 dimension, and close to 10^7 for 10 dimensions.

Metric adaptation.

In our experiments, we use Gaussian kernel shapes. The smoothing parameters are estimated by minimising the leave-one-out (LOO) cross-validation error using the conjugate gradient method with an approximate line search [13]. The comparison between different splits [15] shows that it seems to be an asymptotically reasonable choice. Note that initialisation of the multi-dimensional minimisation does not seem to be an issue here, and we initialise our metric parameters at 20% of the inverse standard deviation on the corresponding dimension. The corresponding bandwidth is around 10 times the sample variance, indicating that we start with a very smooth model. Figure 2 presents the metric parameters h_p^2 obtained for different sample sizes. Several replications of each sample size have been used: 200 replication of 50 samples, 100 replications of

100 samples, 50 replications of 200 samples and 20 replications of 500 samples, so that the total number of generated samples is 10^4 in all cases. All average results display a sizeable attenuation of the metric parameters associated with the 5 irrelevant dimensions, numbered 6 to 10. Recall that small values of the metric parameter correspond to broad kernels, ie average over the corresponding input. Another commendable feature is observed on the first two dimensions: equivalent average metric parameters reflect the symmetric role played by x_1 and x_2.

The standard deviations show that the results obtained with 50 samples are extremely variable, while the high level of the average values suggest that the resulting kernel smoother over-fits the data rather badly. For 100 or more samples, the results associate consistently the first 5 inputs with non-zero metric parameters. The standard deviation on each parameter decreases with larger sample sizes, while the metric parameters tend to increase. This means that the equivalent kernel size becomes smaller as the amount of data increases, a feature that is expected intuitively and asymptotically.

Performance.

Let us now compare the three following models:

- linear ridge regression on 10 inputs, obviously a bad choice for modelling (8),
- kernel regression with one smoothing parameter $\mathbf{H} = h^2\mathbf{I}$,
- adaptive metric kernel with a diagonal smoothing matrix $\mathbf{H} = \text{diag}(h_1^2, h_2^2, \ldots, h_{10}^2)$.

All parameters (ridge, h and h_p) are set by minimising the LOO error. In order to test how the input dimension is effectively reduced by the adaptive metric approach, we will use the last two models on the full input space (5 relevant dimensions + 5 noisy dimensions) as well as on the reduced input space (5 relevant dimensions). The results from table 2 are averages obtained on three dataset sizes containing 100, 200 and 500 samples, with 100, 50 and

20 replications, respectively. As expected, both kernel methods perform better than linear regression on the training set. However, the standard kernel (with 10 inputs) does consistently worse on the test set, while the adaptive metric method performs slightly worse on this particular 100 sample dataset, and clearly better on larger sample sizes. For most kernel methods, the results indicate some kind of over-fitting, while the test error stay well above the noise level of 1. However, as noted above, we have too few data for a reliable estimation of the regression on the basis of neighbouring data points. When modelling from the 10-dimensional input space, adapting the metric produces a rather impressive gain in performance for all sample sizes. A comparison between the results obtained with 10 inputs versus the 5 relevant inputs shows that the standard kernel gains a lot from the removal of the noisy dimensions. On the other hand, the adaptive metric method displays a very limited gain, reflecting the fact that these noisy inputs have been effectively neutralised. Notice that with 200 and 500 samples, the 10 dimensional adaptive metric kernel performs slightly better than the 5 dimensional standard kernel.

5 Benchmarking

In order to evaluate the performance of the adaptive metric approach to kernel regression, we compare it to other methods using the DELVE environment [16, 17]. DELVE (Data for Evaluating Learning in Valid Experiments) is to our knowledge the only currently available environment for performing fully principled comparisons between learning methods. In particular it takes into account training set and test cases effects that are overlooked by standard paired tests on single test sets.

We will use 6 datasets in our comparison:

Boston/NOX The Boston housing dataset [18], predicting the nitric oxides concentration from 13 other attributes, 506 cases in all.

Boston/Price Same as above but predicting the median value of owner-occupied homes.

Pumadyn-8fh Data from a realistic simulation of the dynamics of a Puma 560 robot arm, predicting the angular accelaration of one of the robot arm's links from 8 other attributes. In that data set, the relationship is fairly linear, with a high amount of noise on the output. There are 8192 cases in all.

Pumadyn-8fm Same as above, but with a moderate amount of noise.

Pumadyn-8nh Same as **Pumadyn-8fh**, with a non-linear input-output relationship (instead of almost linear).

Pumadyn-8nm Same as above, but with a moderate amount of noise.

The input dimension is thus 13 for the first two tasks, and 8 for the last 4. For the Boston housing datasets, DELVE splits the data in such a way that there is one common test set containing 250 points, and several non-overlapping training sets of different sizes: 8 training sets of 32 cases, 4 of 64 and 2 of 128. For the Pumadyn family, we consider non-overlapping training sets of sizes 64 to 1024, increasing by powers of two[1]; the 4096 test cases are split in disjoint sets, one per training set for each training set size, in order to have independant test estimates (see the DELVE documentation [16] for details). Note that the amount of data available in the largest training sets is modest compared to the total available data, but this is necessary for providing a principled comparison.

We assess the results obtained by kernel regression using the adaptive metric against 3 standard models from DELVE:

- Linear model with regularisation set by minimising the leave-one-out cross-validation error;
- Gaussian Processes with hyper-parameters set by maximising the posterior probability (MAP) using a conjugate gradient optimisation [17].
- Ensemble of multi-layer Perceptrons (MLP) trained by early stopping [17]: the number of MLP, each trained on a random subset of 2/3 of the available

[1] There are 8 training sets of sizes 64 to 512 and 4 training sets of 1024 cases.

data using early stopping on a validation set containing the remaining 1/3, is limited by a bound on the training time[2].

An additional strength of DELVE is that methods are contributed directly by their authors and resulting losses are saved in the environment. It is thus unnecessary to implement the method again for each new benchmark. In addition, there is a guarantee that the performance of a given reference method is identical from one benchmark to the next. The fact that most methods are contributed by their author is also a guarantee against possible implementation problems when trying to replicate methods from the literature.

The price to pay for these guarantees is that only methods that have been run on the studied data can be used for the comparison. In particular, the ensemble neural network used here does not use Adaptive Relevance Determination [19]. The Gaussian Processes, on the other hand, use one hyper-parameter per input dimension and therefore implement a kind of adaptive metric. The three methods chosen here do therefore provide three different levels of sophistication: 1) the linear model provides a basic reference for competing method; 2) the neural network is a widely used flexible model, which in this implementation uses no adaptive input metric; 3) Gaussian Processes are a fully non-parametric modelling method and implement an input metric adaption that is comparable to our method.

Tables 3 and 4 summarise the results. Note that the differences and significances reported are between the adaptive metric kernel method and each of the three competing models, and that we give no information on the pairwise comparisons between these three methods. In the tables, we have reported in bold the results that appear to be significant at the traditionally accepted level of 5%.

For the Boston datasets, there is only one significant difference, for predicting the nitric oxide concentration using 128 cases, where the adaptive metric kernel seem to significantly outperform the linear model (at the 5% level). None of the

[2] `http://www.cs.utoronto.ca/~delve/methods/mlp-ese-1/hypertext/mlp-ese-1.html` for a full description

other pairwise differences is significant at any reasonnable level. Note that 128 cases is still rather little for an 8 dimensional dataset.

For the Pumadyn family, the differences seem to increase with the non-linearity and decrease with the amount of noise. Predictably, the linear model is largely outperformed on the non-linear dataset (it is also outperformed by the Gaussian Process and neural networks models). The Gaussian Processes yield significantly better performance on the last dataset (non-linear with moderate noise) but otherwise the performance is quite comparable between both methods. Surprisingly, the neural networks ensemble seem to do quite badly when trained on the smallest training sets. This is a surprise because we would expect a local method like kernel regression (and to a lesser extent Gaussian Processes) to be inferior to a projective method like neural networks for small amount of data. However, the differences seem to even out for large training sets.

The residual plots in figures 3 to 5 shed additional light on these results. Each subplot displays a boxplot of the residuals (absolute or squared) obtained from each method. The central box indicates the central 50% values, ie values between the second and third quartiles, the horizontal bar the average, and the vertical dotted bar connect the box to the extreme values (minimum and maximum). In the x-axes labels, LIN stands for linear model, AMK for the adaptive metric kernel approach, GP for the Gaussian Process, and MLP for the multi-layer perceptron. The number under each label indicates the significance level with the same notation as before.

6 Comments

The above experiments show that the adaptive metric kernel is an efficient alternative to standard kernel methods and compares favourably with established flexible modelling techniques such as multi-layer Perceptron or Gaussian Processes. It provides a straightforward non-parametric estimator, requires limited training time to estimate the smoothing parameter, and manages to effectively adapt the input space metric to provide better performance. However, we have

noted that the curse-of-dimensionality, though limited by adapting the metric, is still a major concern for multivariate kernel regression. This is illustrated by the comparatively modest performance of this approach on small sample sizes in the experiments above. Let us also mention that we published in [12] an analysis showing that on a 200 sample dataset obtained from the example in section 4, the performance of a non-linear neural network model (trained with adaptive regularisation) on the test set was 3.01 and even down to 2.26 after retraining.

Our approach is related to other non-parametric techniques in the literature. A similar method was proposed in [20], with a number of key differences. First it addressed the case of classification instead of regression. The focus was also on a *local kernel* method, where the size of the kernel depends on the location in space, as with the nearest-neighbour estimation. Furthermore, only leave-one-out, not general V-fold cross-validation is considered. The last and possibly most significant difference is the parameterisation chosen. We use a single diagonal smoothing matrix, while [20] combines the metric parameters with a local kernel size σ. An undesirable effect of this scheme is that it introduces a redundancy in the parameters, hence an infinite number of equivalent solutions.

The non-parametric estimator (3) is also obtained as the solution of a locally constant regression, weighted by the kernel weights. A natural extension is to consider locally linear or locally polynomial regression. The application of the adaptive metric approach in this context is straightforward, though the required local regression requires a matrix inversion each time we want to obtain a model output, and is thus more computationally demanding than the local constant method. Note also that polynomial regression of order larger than 1 requires the estimation of a number of parameter that is exponential in the input dimension. This can fortunately be tackled by proper regularisation [21].

Finally, instead of focussing on an expression such as (3) that is linear in the output targets $y^{(k)}$, it is possible to swap the role of $K\left(d_{\mathbf{H}}\left(\mathbf{x}, \mathbf{x}^{(k)}\right)\right)$ and $y^{(k)}$ in order to obtain a model that is linear in the Kernel weights. Combined with

a local linear regression, this leads to a model that is very close to the method of Gaussian Processes [22] used eg in the benchmark above.

Matlab code implementing adaptive metric kernel regressions (supporting several kernel shapes and general cross-validation split-ratio) is publicly available through the WWW at the following URL:
`ftp://eivind.imm.dtu.dk/dist/software/matlab/AMKREG` and the DELVE environment is available at `http://www.cs.toronto.edu/~delve/`

7 Summary

In this contribution we consider the well-known non-parametric kernel regression method. We show that in a multivariate setting, the smoothing matrix can be parameterised to reflect the metric of the input space. The cross-validation error associated with a given smoothing matrix is explicitly calculated, as well as its derivative w.r.t. the smoothing parameters. This allows to automatically adapt the metric so as to minimise the estimated average generalisation error.

The performance of this approach is compared to a standard kernel regression approach on an input selection problem. It is shown to limit the input space efficiently by selecting the proper features, and yields improved performance for moderate sample sizes. Additional benchmarking within DELVE show that the performance of the method compares favourably with advanced modelling techniques on the 6 datasets chosen. Challenging prospects for future research include the study of other methods for setting the smoothing parameters, as well as the extension of this scheme to a locally weighted regression.

Acknowledgements.

This work was supported by the Technical University of Denmark, by BIOMED II grant number BMH4-CT97-2775, by the Danish Research Councils through the Danish Computational Neural Network Center (CONNECT) and the THOR Center for Neuroinformatics. JL furthermore acknowledges the Radio Parts Foundation for financial support. CG thanks Carl Edward Rasmussen for dis-

cussions on this method and for help with DELVE.

References

[1] W. Härdle. *Applied nonparametric regression.* Number 19 in Econometric Society Monographs. Cambridge University Press, 1990.

[2] Brian D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[3] H. Schiøler and U. Hartmann. Mapping neural networks derived from the Parzen window estimator. *Neural Networks*, 5(6):903–909, 1992.

[4] Donald F. Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, November 1991.

[5] R. E. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, New Jersey, 1961.

[6] E. A. Nadaraya. On estimating regression. *Theory Probab. Appl.*, 10:186–190, 1964.

[7] G. S. Watson. Smooth regression analysis. *Sankhyā Ser. A*, 26:101–116, 1964.

[8] M.P. Wand and M.C. Jones. *Kernel Smoothing.* Number 60 in Monographs on Statistics and Applied Probability. Chapman & Hall, London, 1995.

[9] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[10] Cyril Goutte. *Statistical learning and regularisation in regression.* Ph.D. thesis 1997/033, Université Paris 6, Paris, july 1997. `http://eivind.imm.dtu.dk/staff/goutte/PUBLIS/thesis.html`.

[11] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36:111–147, 1974. with discussion.

[12] Cyril Goutte and Jan Larsen. Adaptive regularization of neural networks using conjugate gradient. In *Proceedings of ICASSP'98*, volume 2, pages 1205–1208. IEEE, 1998.
`ftp://eivind.imm.dtu.dk/dist/1998/goutte.icassp98.ps.Z`.

[13] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[14] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[15] Cyril Goutte and Jan Larsen. Optimal cross-validation split ratio: Experimental investigation. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *ICANN'98*, volume 2, pages 681–686, Berlin, 1998. Springer Verlag.
`ftp://eivind.imm.dtu.dk/dist/1998/goutte.icann98.html`.

[16] C. E. Rasmussen, R. M. Neal, G. E. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani. *The DELVE manual*, version 1.1 edition, December 1996.
`http://www.cs.utoronto.ca/~delve/`.

[17] Carl Edward Rasmussen. *Evaluation of Gaussian processes and other methods for non-linear regression*. PhD thesis, Department of Computer Science, University of Toronto, 1996.
`http://bayes.imm.dtu.dk/`.

[18] D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *J. Environ. Economics and Management*, 5:81–102, 1978.

[19] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, New York, 1996.

[20] David Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7(1):72–85, January 1995.

[21] Cyril Goutte and Lars Kai Hansen. Regularization with a pruning prior. *Neural Networks*, 10(6):1053–1059, 1997.
`ftp://eivind.imm.dtu.dk/dist/1996/goutte.regularization.ps.Z`.

[22] C. K. I. Williams. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. Technical Report NCRG/97/012, Neural Computing Research Group, Aston University, UK, 1997. http://neural-server.aston.ac.uk/Papers/postscript/NCRG_97_012.ps.Z.

[23] Cyril Goutte and Jan Larsen. Adaptive metric kernel regression. In Tony Constantinides, Sun-Yuan Kung, Mahesan Niranjan, and Elisabeth Wilson, editors, *Neural Networks for Signal Processing VIII – Proceedings of the 1998 IEEE Workshop*, pages 184–193, Piscataway, New Jersey, 1998. IEEE.

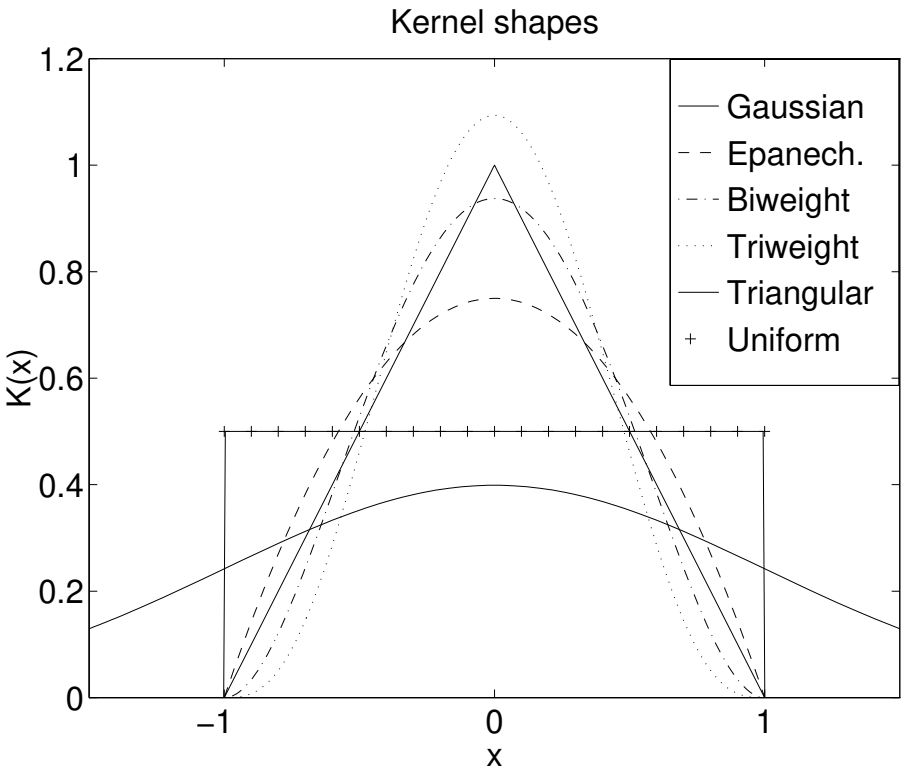


Figure 1: Kernel shapes: 'Epanech.' stands for Epanechnikov (quadratic) kernel.

Kernel	$K(\mathbf{u},\mathbf{v})$	$\frac{\partial K(\mathbf{u},\mathbf{v})}{\partial h_p}$
Gaussian	$\exp\left(-\frac{d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})}{2}\right)$	$-h_p(u_p-v_p)^2\exp\left(-\frac{d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})}{2}\right)$
Epanechnikov	$\left(1-d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})\right)1_{\mathbf{H}}$	$-2h_p(u_p-v_p)^2 1_{\mathbf{H}}$
Biweight	$\left(1-d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})\right)^2 1_{\mathbf{H}}$	$-4h_p(u_p-v_p)^2\left(1-d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})\right)1_{\mathbf{H}}$
Triweight	$\left(1-d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})\right)^3 1_{\mathbf{H}}$	$-6h_p(u_p-v_p)^2\left(1-d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})\right)^2 1_{\mathbf{H}}$
Triangular	$\left(1-\sqrt{d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})}\right)1_{\mathbf{H}}$	$-\frac{h_p(u_p-v_p)^2}{\sqrt{d^2_{\mathbf{H}}(\mathbf{u},\mathbf{v})}}1_{\mathbf{H}}$

Table 1: Kernel expressions and their derivatives w.r.t. the smoothing parameters, using the parameterisation in (4). $1_{\mathbf{H}} = 1_{\{d^2_{\mathbf{H}}<1\}}$ is the indicator function. Note that no scaling is used so that these kernels do not integrate to 1.

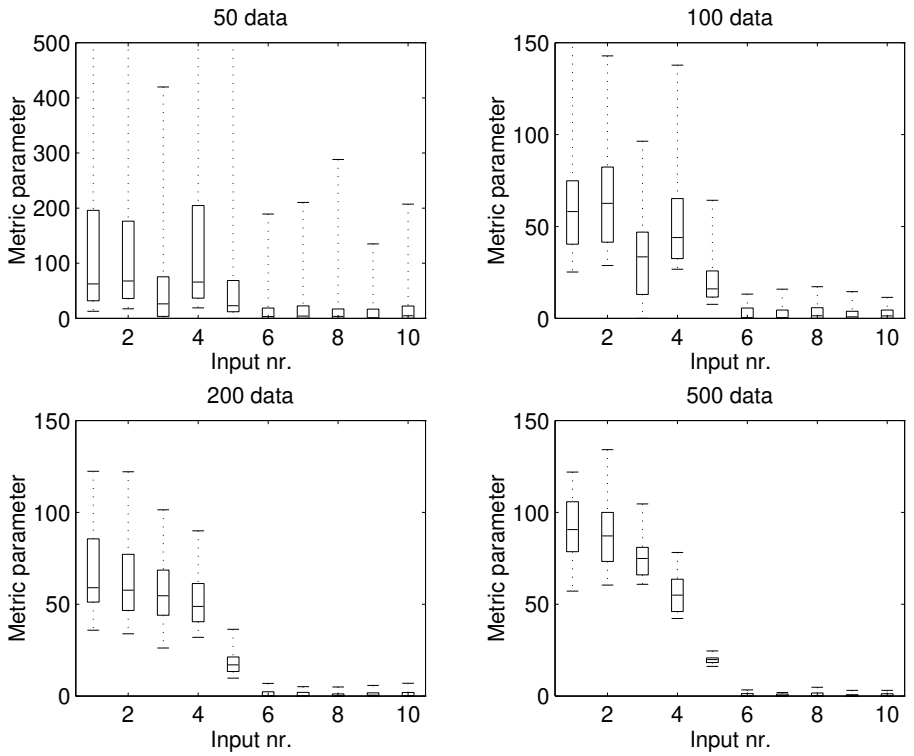


Figure 2: Metric parameters h_p^2, $p \in \{1, \ldots 10\}$, for 50 to 500 samples training sets. The box indicates the central 50%, i.e. the second and third quartiles, the horizontal line is the average and the whiskers indicate the extreme values. Results are abtained from 20 to 200 replicated experiments, so that the total amount of samples is 10000. Notice that all but the top-left plot have the same vertical scale.

Sample size:		100		200		500	
Model	Dim	Train	Test	Train	Test	Train	Test
Linear	10	6.21	7.74	6.33	7.38	6.92	7.19
Standard kernel	10	0.48	11.4	0.42	9.82	0.34	7.84
Standard kernel	5	0.93	6.37	0.78	4.96	0.74	3.44
Adaptive metric	10	0.24	7.30	0.38	**4.87**	0.50	**3.20**
Adaptive metric	5	0.64	6.50	0.60	**4.71**	0.65	**3.17**

Table 2: Average results of the three models for three training set sizes. 'Train' and 'Test' stand for the training and test error (calculated on 5000 samples).

AMK versus			LIN	GP	MLP
Dataset	Size	Loss			
Boston/NOX	32	A	-	-	n/a
Boston/NOX	32	S	-	-	n/a
Boston/NOX	64	A	-	-	n/a
Boston/NOX	64	S	-	-	n/a
Boston/NOX	128	A	**-13 (5)**	-	n/a
Boston/NOX	128	S	-	-	n/a
Boston/Price	32	A	-	-	-
Boston/Price	32	S	-	-	-
Boston/Price	64	A	-	-	-
Boston/Price	64	S	-	-	-
Boston/Price	128	A	-	-	-
Boston/Price	128	S	-	-	-
Pumadyn-8fh	64	A	4.4 (10)	-	4.6 (9)
Pumadyn-8fh	64	S	-	-	-
Pumadyn-8fh	128	A	**2.7 (2)**	-	**3.0 (1)**
Pumadyn-8fh	128	S	**5.1 (3)**	**5.7 (2)**	**5.3 (2)**
Pumadyn-8fh	256	A	-	**-2.6 (1)**	-
Pumadyn-8fh	256	S	-	-	-
Pumadyn-8fh	512	A	**-1.2 (1)**	**-2.5 (0)**	-0.5 (7)
Pumadyn-8fh	512	S	**-1.1 (1)**	**0.7 (5)**	**-0.5 (5)**
Pumadyn-8fh	1024	A	**-1.6 (5)**	**-2.4 (1)**	-
Pumadyn-8fh	1024	S	**-1.6 (4)**	-	-

Table 3: Standardised expected loss difference (in percent) between 3 methods and the adaptive metric approach, for (A)bsolute and (S)quared losses, for increasing sample sizes between 32 and 128. Negative values mean the adative metric is superior. The significance of the difference is in parenthesis and values are only reported if the significance is smaller than 0.1, otherwise '-' is reported. Significance are reported as follows: 10 means $p < 0.1$, 9 means $p < 0.09$, ... 1 means $p < 0.01$ and 0 means $p < 0.001$. n/a means the method was not available for comparison. Accordingly, 5 or less indicates significance at the 5% level (indicated in bold), 1 or less significance at the 1% level.

AMK versus			LIN	GP	MLP
Dataset	Size	Loss			
Pumadyn-8fm	64	A	-	-	-
Pumadyn-8fm	64	S	-	**2.8 (3)**	-
Pumadyn-8fm	128	A	**-2.7 (1)**	-	-
Pumadyn-8fm	128	S	**-1.2 (2)**	**1.4 (3)**	-
Pumadyn-8fm	256	A	**-4.1 (0)**	-	-
Pumadyn-8fm	256	S	**-1.9 (0)**	**0.5 (0)**	-
Pumadyn-8fm	512	A	**-4.6 (0)**	**-0.8 (2)**	-
Pumadyn-8fm	512	S	**-2.1 (0)**	0.4 (9)	-
Pumadyn-8fm	1024	A	**-5.0 (0)**	**-0.9 (3)**	-
Pumadyn-8fm	1024	S	**-2.3 (0)**	**0.3 (3)**	-
Pumadyn-8nh	64	A	**-17.6 (1)**	-	**-12.6 (2)**
Pumadyn-8nh	64	S	**-18.9 (2)**	10.2 (7)	-
Pumadyn-8nh	128	A	**-19.9 (0)**	-	**-7.4 (1)**
Pumadyn-8nh	128	S	**-25.0 (0)**	5.4 (10)	**-7.9 (2)**
Pumadyn-8nh	256	A	**-21.2 (0)**	-	**-2.8 (3)**
Pumadyn-8nh	256	S	**-27.0 (0)**	-	-
Pumadyn-8nh	512	A	**-22.5 (0)**	**-1.8 (4)**	**-2.0 (2)**
Pumadyn-8nh	512	S	**-28.9 (0)**	**2.2 (3)**	**-1.9 (3)**
Pumadyn-8nh	1024	A	**-23.5 (0)**	**-2.3 (3)**	-0.6 (9)
Pumadyn-8nh	1024	S	**-30.0 (0)**	-	**-1.1 (3)**
Pumadyn-8nm	64	A	**-42.8 (0)**	**5.8 (0)**	**-21.6 (0)**
Pumadyn-8nm	64	S	**-44.6 (0)**	**6.0 (0)**	**-18.8 (0)**
Pumadyn-8nm	128	A	**-45.4 (0)**	**2.1 (0)**	**-8.8 (0)**
Pumadyn-8nm	128	S	**-44.5 (0)**	**2.2 (0)**	**-5.4 (0)**
Pumadyn-8nm	256	A	**-47.2 (0)**	**0.9 (1)**	**-0.9 (2)**
Pumadyn-8nm	256	S	**-46.5 (0)**	**1.2 (0)**	-
Pumadyn-8nm	512	A	**-47.7 (0)**	0.6 (6)	-
Pumadyn-8nm	512	S	**-45.9 (0)**	**1.0 (0)**	**0.2 (4)**
Pumadyn-8nm	1024	A	**-48.3 (0)**	-	-
Pumadyn-8nm	1024	S	**-45.6 (0)**	**0.7 (1)**	0.2 (9)

Table 4: See caption in table 3. Note that here the sample sizes are increasing between 64 and 1024.

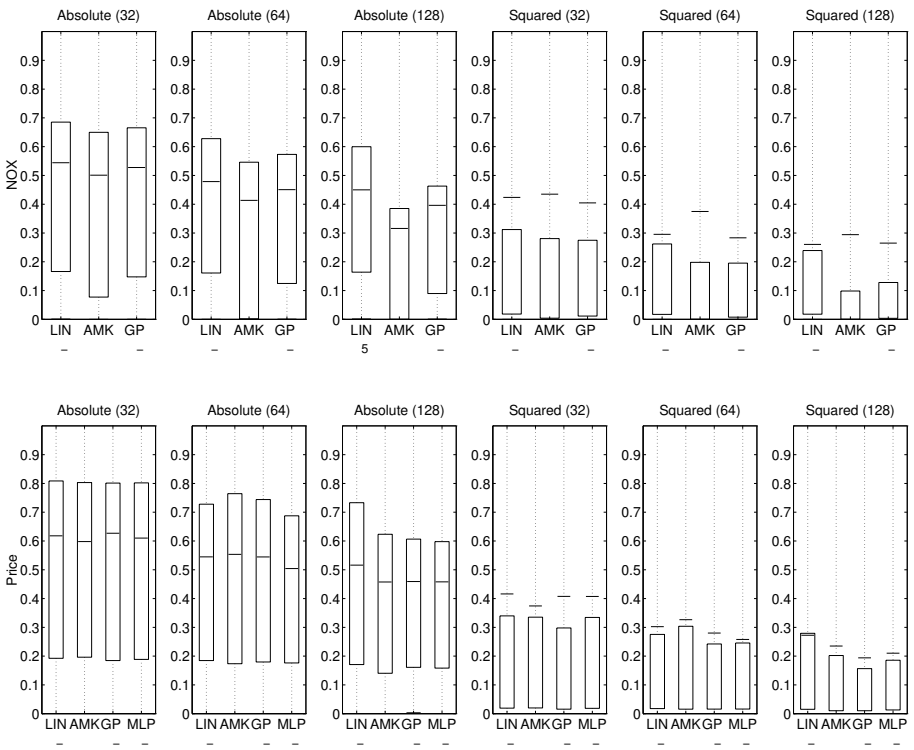


Figure 3: Boxplot of standardised residuals for the Boston housing dataset, predicting the Nitric Oxide concentration (NOX) and median house price (Price). Title of each subplot indicates loss (Absolute or Squared) and training set size (in parenthesis). The x-axes labels stand for: LIN-linear model, AMK-adaptive metric kernel regression, GP-Gaussian Process, MLP-multi-layer perceptron. Number under model label indicates significance of difference with AMK, using the same convention than in tables 3 and 4. Dash (-) indicates no significant difference. For absolute loss, 32 data and the NOX task (top row, third plot), the adaptive metric kernel regression outperforms the linear model significantly at the 5% level. All other difference are not significant. Notice 1) the average squared loss (horizontal bar) in the 3 plots on the right is higher that the third quartile (box upper end), and 2) the maximum is way above the displayed limits, indicating a very skewed distribution.

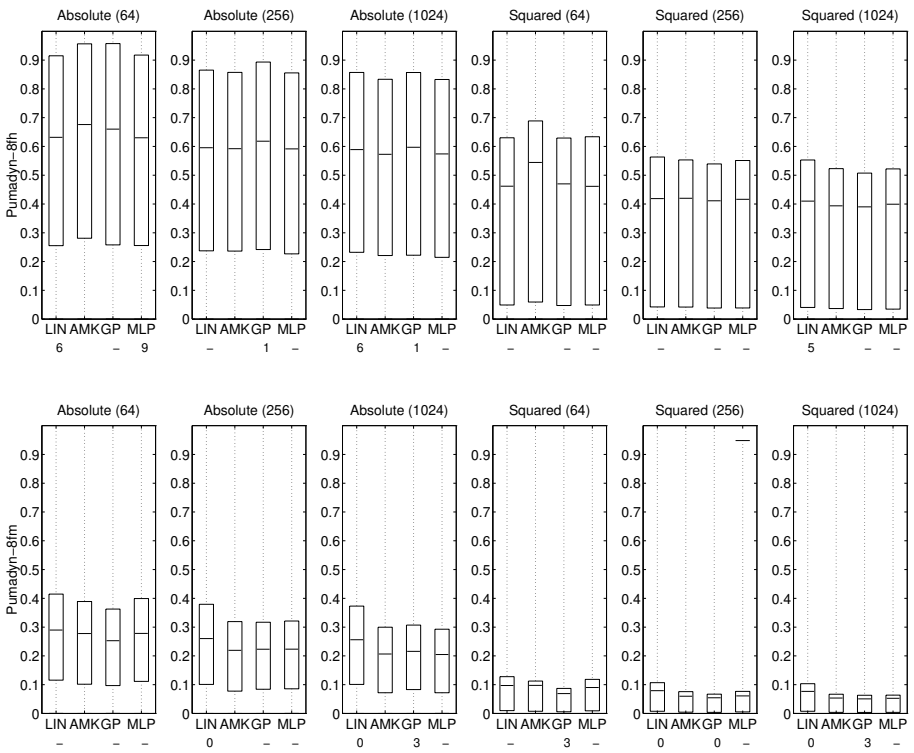


Figure 4: Boxplot of standardised residuals for robotic arm dataset, fairly linear with high (Pumadyn-8fh) and moderate (Pumadyn-8fm) level of noise. Title of each subplot indicates loss (Absolute or Squared) and training set size (in parenthesis). The x-axes labels stand for: LIN-linear model, AMK-adaptive metric kernel regression, GP-Gaussian Process, MLP-multi-layer perceptron. Number under model label indicates significance of difference with AMK (cf. tables 3 and 4). Dash (-) indicates no significant difference. For example, for squared loss, 256 data on the 8fm task (bottom row, fifth plot), the adaptive metric kernel regression outperforms the linear model significantly at the 0.1% level, but is significantly outperformed by the Gaussian Processes at the same level.

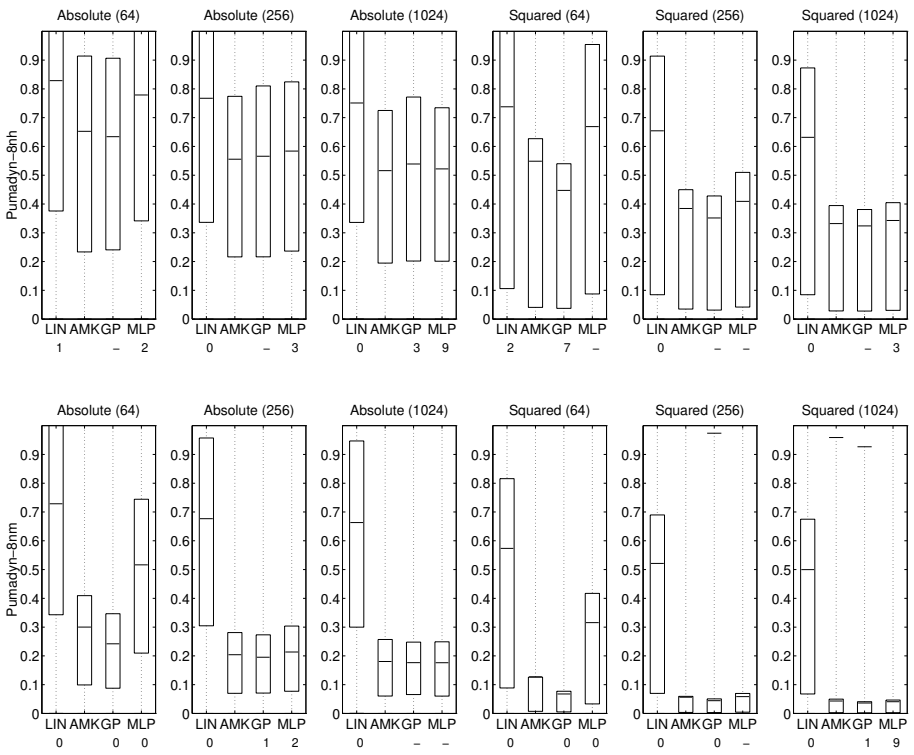


Figure 5: Boxplot of standardised residuals for robotic arm dataset, non-linear with high (Pumadyn-8fh) and moderate (Pumadyn-8fm) level of noise. Title of each subplot indicates loss (Absolute or Squared) and training set size (in parenthesis). The x-axes label stand for: LIN-linear model, AMK-adaptive metric kernel regression, GP-Gaussian Process, MLP-multi-layer perceptron. Number under model label indicates significance of difference with AMK (cf. tables 3 and 4). Dash (-) indicates no significant difference. For example, for absolute loss, 64 data on the 8nm task (bottom row, first plot), the adaptive metric kernel regression outperforms both linear model and MLP significantly at the 0.1% level, but is significantly outperformed by the Gaussian Processes at the same level.