Hessian Corrected Input Noise Models

Botond Attila Bócsi and Lehel Csató

Faculty of Mathematics and Informatics, Babeş-Bolyai University
{bboti,lehel.csato}@cs.ubbcluj.ro

Abstract. When the inputs of a regression problem are corrupted with noise, integrating out the noise process leads to biased estimates. We present a method that corrects the bias caused by the integration. The correction is proportional to the Hessian of the learned model and to the variance of the input noise. The method works for arbitrary regression models, the only requirement is two times differentiability of the respective model. The conducted experiments suggest that significant improvement can be gained using the proposed method. Nevertheless, experiments on high dimensional data highlight the limitations of the algorithm.

1 Introduction

In regression problems we find the optimal mapping that explains the relationship between an input space and an output space. Along with the class from where the model is taken, regression methods assume the presence of noise in the data. Most regression methods assume output noise but neglect the possibility of the input noise due to analytical intractability or the fact that for simple linear models the input noise is transferred to the output. In this paper, we analyse regression methods with input noise assumption.

Input noise modelling is more difficult than modelling output or observational noise: whilst the output noise is directly observable, the input noise can be observed only through the input-output transformation that we *inferred*. In many cases there are analytically tractable solutions for the output noise model, while such solutions are extremely rare for the input noise, a notable exception is the restrictive linear model. Furthermore, even when the input noise can be integrated out analytically, the solution will be biased. As shown by [1], adding noise to the inputs can be used for regularization, but for non-zero curvatures, the estimated output is biased.

Consider for example the parabola in Figure 1, where the red circles on the horizontal axis are the noisy measurement locations in the neighbourhood of zero – Gaussian noise is assumed –, the blue stars are the regression values of the measurements – without output noise for clarity reasons – and the green point is the result of the Monte Carlo approximation of the integration. It is clear that after integration the prediction at zero will be always below the true value of the function, caused by the non-zero curvature of the function, e.g., for concave functions the averaging over the noisy measurements *pulls down* the prediction.

V. Mladenov et al. (Eds.): ICANN 2013, LNCS 8131, pp. 1–8, 2013.

The method proposed in this article improves regression with input noise by removing the bias induced by the input noise. We show that the bias is proportional to the variance of the noise and to the second order derivative (Hessian) of the objective function. The method can be used for any regression algorithm as long as its Hessian can be calculated.

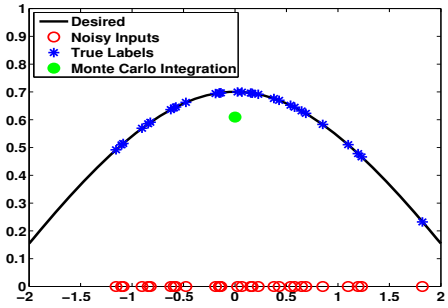


Fig. 1. The bias when estimating with input noise: the predicted value at zero will *always* be underestimated

1.1 Related Work

Most of the work with input noise has been done for specific models, e.g., Gaussian processes (GP), neural networks (NN), that we discuss later. An exception is [11] who showed that the regularization with input noise [1] is equivalent to adding the L_2 norm of the Hesse matrix of the objective function. Our method is based on this insight with reversed goals, i.e., we do not want more regularized models but better accuracy. In the GP framework a general idea is to use a second GP to model the input noise process. Posteriors have been obtained via Monte Carlo integration [6], variational learning [8], or EM-based methods [7]. Another approach is to use the Taylor expansion of the GP [5] and analytically compute the posterior (e.g., for squared exponential kernels) [2]. The integration of the input noise is intractable for NNs as well. To approximate the posterior, [12] integrated over the uncertain input with a maximum likelihood estimation, while [13] used Laplace approximation and Monte Carlo simulation to improve on the prediction.

The above approaches do not report improvement in the accuracy of the prediction, they focus instead on improved posterior variance estimates. Furthermore, most of them were applied for one dimensional problems, with only a few being used on real world data-sets with multidimensional inputs, e.g., [2,9,8]. We apply our method for both artificial problems and real-world data-sets.

2 Input Noise Correction

Let us be given a data-set $\boldsymbol{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, with inputs $\boldsymbol{x}_i \in \mathfrak{R}^d$ and labels $y_i \in \mathfrak{R}$. A general assumption is that the labels are corrupted with additive Gaussian noise, and we also assume that the inputs are also corrupted, i.e.,

$$y = \tilde{y} + \epsilon_y \qquad\qquad \boldsymbol{x} = \tilde{\boldsymbol{x}} + \boldsymbol{\epsilon}_x,$$

where y is the observed label, $\tilde{y}$ is the true label, and $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ is a noise process; $\boldsymbol{x}$ is the observed input, $\tilde{\boldsymbol{x}}$ is the true input, and $\boldsymbol{\epsilon}_x \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ is the additive Gaussian input noise. We assume that the inputs are uncorrelated, i.e., $\boldsymbol{\Sigma}$ is a diagonal covariance matrix with σ_i^2 on the diagonal and $\boldsymbol{\sigma}$ denotes the

vector of individual variances. If we denote with $f(\cdot)$ the true data generating function, with $p(\boldsymbol{\epsilon}_x)$ the distribution of the input noise, and with $\hat{f}(\cdot)$ the function after integrating out the input noise, the relation between them is

$$\hat{f}(\tilde{\boldsymbol{x}}) = \int f(\tilde{\boldsymbol{x}} + \boldsymbol{\epsilon}_x)\, dp(\boldsymbol{\epsilon}_x), \tag{1}$$

The estimate $\hat{f}(\cdot)$ is biased even when the true generating function $f(\cdot)$ is known [12]. We propose to use the second order Taylor expansion of $f(\cdot)$ around the true input location $\tilde{\boldsymbol{x}}$, with the averaging as

$$\hat{f}(\tilde{\boldsymbol{x}}) = \int \left(f(\tilde{\boldsymbol{x}}) + \boldsymbol{\epsilon}_x^\top J_f(\tilde{\boldsymbol{x}}) + \frac{1}{2}\boldsymbol{\epsilon}_x^\top H_f(\tilde{\boldsymbol{x}})\boldsymbol{\epsilon}_x + \dots \right) dp(\boldsymbol{\epsilon}_x), \tag{2}$$

where $J_f(\boldsymbol{x})$ and $H_f(\boldsymbol{x})$ are the Jacobian and the Hessian of $f(\boldsymbol{x})$. The first term does not depend on the noise; the Jacobian term vanishes since $\boldsymbol{\epsilon}_x$ is has zero mean; and the third term can be written as $\boldsymbol{\epsilon}_x^\top H_f(\tilde{\boldsymbol{x}})\boldsymbol{\epsilon}_x = \mathrm{tr}(H_f(\tilde{\boldsymbol{x}})\, \boldsymbol{\epsilon}_x \boldsymbol{\epsilon}_x^\top)$, leading to the following simplified expression:

$$\hat{f}(\tilde{\boldsymbol{x}}) \simeq f(\tilde{\boldsymbol{x}}) + \frac{1}{2}\boldsymbol{\sigma}^\top H_f(\tilde{\boldsymbol{x}})\boldsymbol{\sigma} \simeq f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\sigma}^\top H_f(\boldsymbol{x})\boldsymbol{\sigma}, \tag{3}$$

The true input location $\tilde{\boldsymbol{x}}$ is unknown; we approximate it with the noisy location $\boldsymbol{x}$, i.e. $f(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}})$ and $H_f(\boldsymbol{x}) = H_f(\tilde{\boldsymbol{x}})$. A similar assumption has been made by [9] in the context of input noise GPs.

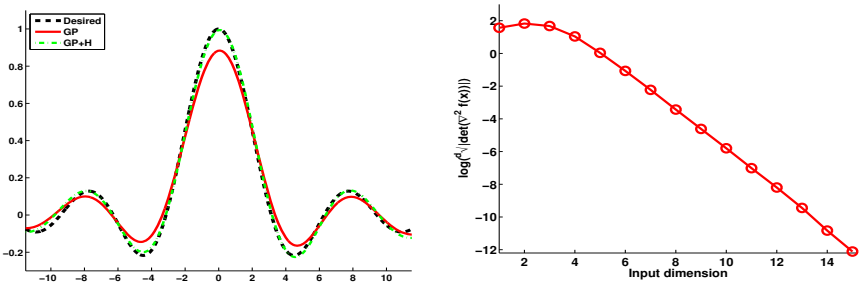
We consider two steps in approximating $\hat{f}(\tilde{\boldsymbol{x}})$: we first construct a function $g(\cdot)$ based on $\boldsymbol{D}$ *without* the input noise assumption. In this first step we do not make specific assumptions about the function, we assume that it can be arbitrary. From Equation (3) follows that $g(\cdot)$ and the *true* data generating function $f(\cdot)$ are related as follows:

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\sigma}^\top H_f(\boldsymbol{x})\boldsymbol{\sigma}. \tag{4}$$

The second step is obtaining $f(\cdot)$ when $g(\cdot)$ and $\boldsymbol{\sigma}$ are known, i.e., solving the partial differential equation from Equation (4). This equation does not have an analytical solution [3] since it requires an integration over $g(\boldsymbol{x})$ that is intractable in most cases. A possible solution is to use numerical methods but these methods time consuming and become unstable when noise is present. A more significant drawback is the lack of good initial conditions for the differential equation. Note that the initial conditions must contain both the values and the derivatives of the function $f(\cdot)$ [3]. We tried the following approximations for the initial conditions

$$f(\boldsymbol{x}) = g(\boldsymbol{x}),\;\; J_f(\boldsymbol{x}) = J_g(\boldsymbol{x}) \qquad \text{or} \qquad f(\boldsymbol{x}_j) = y_j,\;\; J_f(\boldsymbol{x}_j) = J_g(\boldsymbol{x}_j),$$

where $(\boldsymbol{x}_j, y_j) \in \boldsymbol{D}$ but our experiments show that pure results can be obtained based on these approximations. Next, we make further assumptions about $f(\cdot)$ to approximate the solution of the partial differential equation (4).



(a) Example for $f(x) = \mathrm{sinc}(x)$. (b) Curvature of the prediction.

Fig. 2. (a) Approximating with 800 training points and input noise with standard deviation $\sigma = 0.8$ the standard GP (red) under- and overestimates the true prediction (black) where the curvature is high. Our GP model corrected with the Hessian (green) results in more accurate prediction. (b) The curvature of the predicted function tends to zero exponentially when the dimension of the inputs increases.

2.1 Quadratic Approximation of the Partial Differential Equation

We approximate $f(\cdot)$ with a quadratic function at every input location $\boldsymbol{x}$, i.e. we assume that $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}_x \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{b} + c$ with its Hessian $H_f(\boldsymbol{x}) = 2\boldsymbol{A}_x$. This is again similar to the approximation based on the Taylor series expansion from Equation (2), and we assume that the expansion is at the current point of interest $\boldsymbol{x}$.

We substitute the local approximation into Equation (4), differentiate it two times and obtain

$$H_g(\boldsymbol{x}) = 2\boldsymbol{A}_x. \tag{5}$$

i.e. under the locally quadratic approximation of $f(\cdot)$, the Hessian of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ *must be equal*. Therefore, we can replace $H_f(\boldsymbol{x})$ with $H_g(\boldsymbol{x})$ from Equation (4), and obtain the following expression for $f(\boldsymbol{x})$

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) - \frac{1}{2}\boldsymbol{\sigma}^\top H_g(\boldsymbol{x})\boldsymbol{\sigma} \tag{6}$$

On the right side of Equation (6) every term is known, thus $f(\boldsymbol{x})$ has an analytic form. The interpretation of Equation (6) is the following: when dealing with data corrupted with input noise, any regression model $g(\cdot)$ can be improved by subtracting the Hessian of the model $H_g(\cdot)$ multiplied by the noise variance. An other interpretation of the result in Equation (6) is that if we relax our assumptions about the model, then we might replace the second derivatives of the true function from Equation (4) with the approximating function $g(\cdot)$ and this approximation is pursued in the rest of the paper.

3 Experiments

We conducted experiments to see how well the proposed method performs under different conditions. We were interested how well it scales with the number of the training examples, with the variance of the input noise, and with the dimension of the input space. We did not compare the proposed method with the state-of-the-art input noise models since the authors do not report significant improvement on the accuracy of the prediction, rather they enhance the posterior variance that is not desired in our framework.

3.1 Illustration for the Sinc(x) Function

We applied the proposed method on a toy example, to give an insight about the induced improvement. For training inputs we generated 800 points on the $[-12, 12]$ interval corrupted with additive Gaussian noise with standard deviation $\sigma = 0.8$. For training labels we transformed these point with the $\text{sinc}(\cdot)$ function and added a Gaussian noise with standard deviation $\sigma_y = 0.1$ as output noise. We used standard GPs to obtain prediction and also used the Hessian corrected version (GP+H). Results are shown on Figure 2a. It can be seen that the standard GP under- or overestimates the true $\text{sinc}(\cdot)$ function where the curvature of the function is high. On the other hand, the Hessian corrected GP results in an almost perfect prediction. Note that when the variance of the input noise is not known, we can under- or over- correct the model using wrong values.

3.2 Synthetic Data

We generated synthetic data for different one dimensional functions, with results in Figures 3. We used GPs and NNs for the regression models to learn these functions and compared the accuracy with the Hessian corrected versions of the respective methods (referred as GP+H and NN+H). The hyper-parameters of the GP were obtained with evidence maximization while the parameters of the NN were obtained using back-propagation.

For every function we investigated (1) how the improvement of the input noise correction scales with increasing the standard deviation of the input noise, and (2) how the improvement of the input noise correction scales with increasing the number of the training examples with a fixed standard deviation. As a measure of performance we used the mean square error (MSE) of the learned function on the same interval where the training data was generated.

For the first type of experiments (first and third columns of Figures 3) we generated 200 points from the interval $[-4, 12]$ (or with the intersection where the respective function was defined) and added a Gaussian output noise with standard deviation $\sigma_y = 0.1$. The standard deviation of the input noise was between 0.1 and 1.2. For the second type of experiments (second and fourth columns of Figures 3) we generated points from the same interval as before, added a Gaussian input noise with standard deviation $\boldsymbol{\sigma} = 0.8$, and also added Gaussian output noise with standard deviation $\sigma_y = 0.1$. The number of the

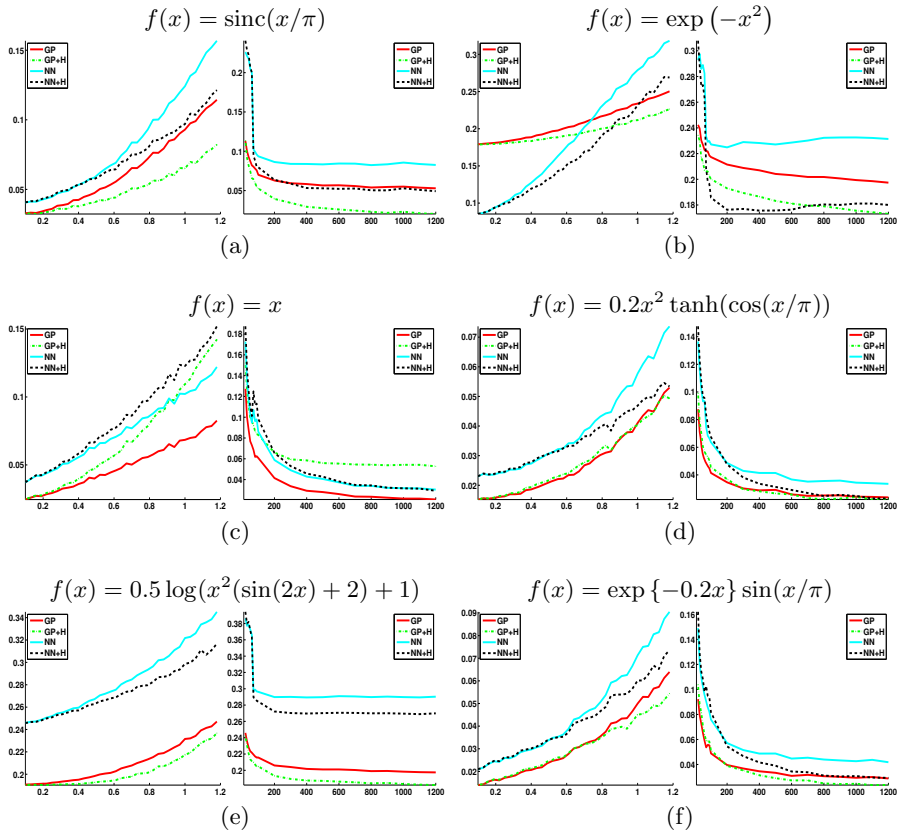


Fig. 3. Performance on artificial data: the left-side shows the evolution of the mean-square error when increasing gradually the input noise level. Plots on the right side show the errors when the size of the available data increases.

training points was between 20 and 1200 and all the shown results are averages over 200 runs.

Synthesizing plots of the algorithm, in Figure 3, show that the Hessian corrected version are almost always better than the standard GP or NN or at least it is very close. There is one exception: the linear function from Figure 3.(c), where the Hessian corrected version is significantly worse. The explanation is that the true Hessian of a linear model is zero. Thus, small inaccuracies in the standard regression model lead to non-zero Hessian, and therefore the prediction will deteriorate.

The general trend with increasing the noise level is that as the standard deviation of the input noise increases, the improvement induced by the Hessian corrected methods is more significant.

Another important conclusion is that as the number of the training examples grows, the MSE of the Hessian corrected estimation decreases (an exception is again the linear function), thus, it is consistent in the sense that it converges to the best model that the chosen function space contains.

Table 1. Results of experiments on real world data-sets (performance measured in MSE). Boston housing ($1000s); Concrete (mega-pascal); Barrett WAM (millimetres); CPU performance (benchmark points); Auto MPG (miles per gallon).

Data-set name	GP	GP+H	NN	NN+H	Noise (σ)	Dim.	Set size
Boston housing ($1000s)	2.2271	2.2271	3.4819	3.4819	0.1	13	506
Concrete (MPa)	4.1281	4.1280	6.1865	6.1864	1	8	1030
Barrett WAM 1 (mm)	2.9272	2.9242	2.8726	2.8488	0.01	4	1000
Barrett WAM 2 (mm)	13.707	13.731	12.439	9.3524	0.1	4	1000
CPU performance	17.762	17.763	16.846	16.846	5	7	209
Auto MPG (mpg)	4.0301	4.0322	2.2990	2.2988	3	7	301

3.3 Real World Data-Sets

When trying the method for real data, we were interested in how significant the Hessian corrected improvement is on higher-dimensional data where there is no control over the noise of the function to be predicted either. The data-sets were gathered from different domains with different features (input dimension, training set size) – Table 1 summarizes the data-sets. We used the (1) Boston housing data-set [4] that concerns housing values in suburbs of Boston; (2) Concrete data-set [14] that collected the compressive strength of the concrete; (3) Barrett whole arm manipulator (WAM) data-set [10] that was generated by us on a simulated Barrett WAM robot architecture while we learned the forward kinematics of the robot arm [10]; (4) CPU performance data-set [4] that deals with the CPU performance; (5) Auto MPG data-set [4] that collected fuel consumption of cars.

Results from Table 1 show the standard deviation of the input noise, the dimension of the data-set, and the size of the data-set as well. The values were obtained with 10-fold cross validation and are averages over 100 runs.

The improvement of the Hessian corrected methods is insignificant but it is generally not worse. We believe that the explanation is the same as it was for the linear functions in the previous section. In high dimensions we prefer rather linear models (e.g., as a result of regularization) to avoid over-fitting. In theory we prefer models with Hessians close to zero. Thus, the addition of the approximated Hessian does not improve the prediction.

To illustrate this effect of high dimensional data on the Hesse matrix, we experimented on a toy example. We approximated a d dimensional parabola $f(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2$ using a GP. We generated 100 points uniformly distributed on the interval $[-2, 2]^d$ and did not add any noise. Figure 2b shows that the curvature of the prediction function tends to zero exponentially when the dimension of the inputs increases. Note that this phenomena is independent of any of our assumptions, it is rather a property of high dimensional data modeling.

4 Discussion

When the inputs of a data-set are corrupted with noise, integrating out the noise process leads to biased estimates. We presented a method that corrects this bias.

The correction is proportional to the Hessian of the learned model and to the variance of the input noise. The method works for arbitrary regression models.

The proposed method has limitations: it does not improve prediction for high-dimensional problems, where the data are *implicitly* scarce. This is due to the fact that the estimated Hessian is considerably flattened, leading to no significant contribution to the overall output. To wisely choose when the Hessian correction can be used with success, these limitations have to be taken into account.

An interesting further research direction is to further analyse our algorithm specialised for the robotic Barrett WAM data. We believe that there are potential improvement capabilities since the size of the data-set is large, the dimensionality of the problem is reduced, and there is a real need for better approximation methods. One could – for example – start from the approximation to the second order PDE from Equation (6) and try to provide still approximating solutions that would probably be more precise than the simple replacement of the true function with its approximated based on noiseless inputs.

The authors acknowledge the support of the Romanian Ministry of Education, grant PN-II-RU-TE-2011-3-0278.

References

1. Bishop, C.M.: Training with noise is equivalent to Tikhonov regularization. Neural Computation 7(1), 108–116 (1995)
2. Dallaire, P., Besse, C., Chaib-draa, B.: An approximate inference with Gaussian process to latent functions from uncertain data. Neuroc. 74(11), 1945–1955 (2011)
3. Evans, L.: Partial Differential Equations. Graduate Studies in Mathematics. American Mathematical Society (2010)
4. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
5. Girard, A., Murray-Smith, R.: Learning a Gaussian process model with uncertain inputs. Tech. rep., University of Glasgow, Department of Computing Science (2003)
6. Goldberg, P.W., Williams, C.K.I., Bishop, C.M.: Regression with input-dependent noise: A Gaussian process treatment. In: NIPS (1997)
7. Kersting, K., Plagemann, C., Pfaff, P., Burgard, W.: Most likely heteroscedastic Gaussian process regression. In: ICML, pp. 393–400. ACM, New York (2007)
8. Lzaro-gredilla, M., Titsias, M.K.: Variational heteroscedastic Gaussian process regression. In: ICML, pp. 841–848. ACM (2011)
9. McHutchon, A., Rasmussen, C.E.: Gaussian process training with input noise. In: NIPS, pp. 1341–1349 (2011)
10. Nguyen-Tuong, D., Peters, J.: Incremental online sparsification for model learning in real-time robot control. Neurocomputing 74(11), 1859–1867 (2011)
11. Rifai, S., Glorot, X., Bengio, Y., Vincent, P.: Adding noise to the input of a model trained with a regularized objective. CoRR abs/1104.3250 (2011)
12. Tresp, V., Ahmad, S., Neuneier, R.: Training neural networks with deficient data. In: NIPS, pp. 128–135. Morgan Kaufman Publishers (1994)
13. Wright, W.: Neural network regression with input uncertainty. In: Neural Networks for Signal Processing VIII, pp. 284–293. IEEE (1998)
14. Yeh, I.C.: Modeling of strength of high performance concrete using artificial neural networks. Cement and Concrete Research 28(12), 1797–1808 (1998)