Anisotropic Noise Injection for Input Variables Relevance Determination
TNN A750

Yves Grandvalet

Abstract

There are two archetypal ways to control the complexity of a flexible regressor: subset selection and ridge regression. In neural networks jargon, they are respectively known as pruning and weight decay. These techniques may also be adapted to estimate which features of the input space are relevant for predicting the output variables. Relevance is given by a binary indicator for subset selection, and by a continuous rating for ridge regression.

This paper shows how to achieve such a rating for a multi-layer perceptron trained with noise (or jitter). Noise injection is modified in order to penalize heavily irrelevant features. The proposed algorithm is attractive as it requires the tuning of a single parameter. This parameter controls the complexity of the model (effective number of parameters) together with the rating of feature relevances (effective input space dimension). Bounds on the effective number of parameters support that the stability of this adaptive scheme is enforced by the constraints applied to the admissible set of relevance indices. The good properties of the algorithm are confirmed by satisfactory experimental results on simulated data sets.

Keywords

Noise injection, Variable selection, Penalization, Hierarchical priors, Automatic relevance determination.

I. INTRODUCTION

When estimating a regression function, the problem of selecting relevant variables is difficult. Optimal subset selection is attractive as it yields simpler and more interpretable models, but it is a combinatorial and acknowledged unstable procedure [1]. In some problems, it may be better to resort to stable procedures penalizing irrelevant variables. This paper introduces such a procedure, based on the noise injection heuristic.

Noise injection (NI) consists in deliberately adding an artificial noise to the input patterns during the training of a multi-layer perceptron (MLP). Before each presentation of an input vector $\boldsymbol{x}_i$, a random noise $\boldsymbol{\eta}$ is added, while keeping the target outputs $\boldsymbol{y}_i$ unchanged. Asymptotically, as the number of presentation of the sample increases, NI is virtually equivalent to minimize the cost C_{NI}:

$$C_{\mathrm{NI}}(\boldsymbol{f}) = \mathbb{E}_{\eta}\left[\frac{1}{\ell}\sum_{i=1}^{\ell} l(\boldsymbol{f}(\boldsymbol{x}_i+\boldsymbol{\eta}),\boldsymbol{y}_i)\right] \quad , \tag{1}$$

where l denotes the loss function, and $\boldsymbol{f}$ is the network function. This heuristic may be used for three different purposes, namely:

1. learn invariance. In this case the added "noise" is a means to enhance the training set, by adding examples of inputs transformed under a known invariance group [2];
2. improve the network robustness regarding inputs inaccuracy [3];
3. control the complexity of oversized networks. Overfitting is avoided by introducing an induction bias. Intuitively, NI can be interpreted as a means to favor solutions such that little changes in $\boldsymbol{x}$ result in little changes in $f(\boldsymbol{x})$ [4], [5], [6], [7].

In cases 1 and 2, the noise distribution is specified by the knowledge of either the invariance group or the robustness required. In the last case, there is no obvious choice for the noise distribution. Usually,

The author is with Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne, B.P. 20.529, 60205 Compiègne Cedex, France. Email: Yves.Grandvalet@hds.utc.fr.

it is arbitrarily set to a multivariate Gaussian: $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d)$, where $\boldsymbol{I}_d$ is the identity matrix on the d-dimensional input space. The variance σ^2 is the hyper-parameter controlling the complexity of the network, and is tuned by minimizing an estimate of generalization error.

When NI is not used, training is performed thanks to the pieces of information given by each pair of the sample: "the target output $\boldsymbol{y}_i$ is observed *at location* $\boldsymbol{x}_i$". With NI, the former is turned to: "the target output $\boldsymbol{y}_i$ is observed *in the neighborhood* of $\boldsymbol{x}_i$". With a Gaussian noise of covariance matrix $\sigma^2 \boldsymbol{I}_d$, spherical neighborhoods are defined. The network is thus asked to perform a smooth mapping in an isotropic input space, and each input variable is supposed to have the same relevance. If a variable is actually irrelevant, one expect to get a constant function in the corresponding direction of the input space. However, as the training is performed on a finite sample, a spurious dependency may be detected.

This paper addresses the issue of a different parameterization of the covariance matrix, with the aim of determining relevant inputs. Anisotropic Noise Injection (ANI) uses a diagonal noise covariance matrix which elements are individually adapted. This parameterization yields elliptic neighborhoods, thus inducing a bias in favor of mappings which are smoother in the directions corresponding to large variances. The relevance of a variable can then be rated by the inverse of the corresponding standard deviation. The ellipsoid can be degenerated in one or several directions (cylinder), in which case the mapping is required to be constant in these directions whose relevances are zero.

In section II, the rationale leading to the algorithm is given for the simple case of a linear perceptron trained with the quadratic loss. The analogy with weight decay shows how NI can be modified in order to determine relevant variables. The sketch of the algorithm is then given in section III, and some of its properties are also discussed for linear and non-linear perceptrons in section IV. Details regarding implementation are given in section V, before the exposure in sections VI and VII of experimental results regarding generalization performances and the ability to rate relevances of input variables. Finally, some comments and discussion conclude the paper.

II. RATIONALE

In this section we consider linear perceptrons with a single output to lighten exposure and notation. Multiple outputs sets can be treated along the same way. The generalization to multi-layer perceptron is the application of the same principles on a more complex model, similar to the extension of linear ridge regression to weight decay.

A. EQUIVALENCE WITH RIDGE REGRESSION

For a linear perceptron $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + \theta$, trained with the quadratic loss l, NI is asymptotically equivalent to minimize the following penalized cost [8], [9]:

$$C_{\mathrm{NI}}(f) = \frac{1}{\ell} \sum_{i=1}^{\ell} \left\| \boldsymbol{w}^T \boldsymbol{x}_i + \theta - \boldsymbol{y}_i \right\|^2 + \boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w} \quad , \tag{2}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the noise $\boldsymbol{\eta}$. For $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_d$, the second right-hand term of (2) is the classical penalizer of ridge regression, or weight decay $\sigma^2 \|\boldsymbol{w}\|^2$.

If the residuals are assumed to be Gaussian, ridge regression can be interpreted in terms of maximum *a posteriori* estimation. The average of residuals is proportional to the log-likelihood and the sum of weights is proportional to the log of the weights prior distribution. The prior is a zero mean Gaussian distribution with diagonal covariance matrix $s^2 \boldsymbol{I}_N$, where s^2 is proportional to $1/\sigma^2$, and N is the number of weights. In this framework, the simplest procedure consists in minimizing C_{NI} (2) for several values of σ^2 or equivalently s^2. This produces a sequence of weights estimates indexed by s^2. Some estimate of generalization error is then computed for each value of the index s^2, and its minimizer is finally selected.

B. AUTOMATIC RELEVANCE DETERMINATION

If ridge regression is used with the intent to determine relevant variables, each weight w_j of $\boldsymbol{w}$ should be penalized differently. This principle is used by MacKay and Neal [10] in the empirical Bayesian framework by the automatic relevance determination (ARD) model. The priors on w_j are of the form $\mathcal{N}(0, s_j^2)$. As the relevance of each input is *a priori* unknown, the s_j^2 should be adapted during training. MacKay and Neal [10], consider two types of prior for s_j^2:
1. s_j^2 are given independent "rather vague" priors;
2. a second level hyper-parameter common to all s_j^2 is given a "very vague" prior, while the s_j^2 themselves are given "less vague" priors, with prior means determined by the common high-level parameter.
In [10], test results for the two versions of ARD are reported on simulated and real data sets. As expected, ARD improves the generalization ability when some inputs are irrelevant. On the three simulated data sets, ARD was beneficial, or at least not detrimental, even when all inputs where relevant. The algorithm failed for only one real data set. Neal reports that it might be due to a stabilization problem in the Markov Chain Monte Carlo scheme used for training.

C. MAP ESTIMATES OF RELEVANCE

In the more rudimentary maximum *a posteriori* estimation framework, the two ARD versions are respectively transposed as follows:
1. minimize (2) with respect to $\boldsymbol{w}$ for several values of $\{s_j^2\}_{j=1}^d$ to produce a sequence of weights multi-indexed by $\{s_j^2\}_{j=1}^d$. Select one element of the sequence by finding the set $\{s_j^2\}_{j=1}^d$ minimizing some estimate of generalization error.
2. minimize (2) with respect to $\boldsymbol{w}$ and $\{s_j^2\}_{j=1}^d$, while a second level hyper-parameter applies some constraint on $\{s_j^2\}_{j=1}^d$. This produces a sequence of weights estimates indexed by this second level hyper-parameter. The value of the index is then computed by minimizing some estimate of generalization error.
In the classical paradigm, the estimate of generalization error is computed for regularly spaced hyper-parameter values, and is then minimized by selecting the best one among all trials. Hence, the first approach requires intensive computation for multiple hyper-parameters, for which trials should be completed over a regular d-dimensional grid of values. Larsen *et al.* [11] addressed this problem by optimizing the estimate of generalization error with respect to the hyper-parameters by gradient descent. They use cross-validation residuals to estimate the generalization error. Orr [12], on a similar problem, optimizes multiple hyper-parameters on an analytical estimate of generalization error (Generalized cross-validation) by an iterative technique. The reported results in [11], [12] show the feasibility of these approaches. However, relying on the optimization of many hyper-parameters over generalization error estimates seems risky. The optimized criteria become down-biased and overfitting might occur.

In the second solution advocated here, the estimate of generalization error is minimized with respect to a single (second level) hyper-parameter, which prevents the priors from vanishing (*i.e.* getting non-informative). Without constraints on $\{s_j^2\}_{j=1}^d$, this solution would be reached, since $s_j^2 \to \infty$, $j = 1, \ldots, d$ sets the second (non-negative) term of (2) to zero, while the first term is then be minimized with respect to $\boldsymbol{w}$, yielding the ordinary least squares estimate.

III. ALGORITHM

One may think of several meaningful constraints to be applied to $\{s_j^2\}_{j=1}^d$. The one proposed here controls the average weight variance: $\frac{1}{d}\sum_{j=1}^d s_j^2 = s^2$, where s^2 is the second level hyper-parameter. This constraint implies that the weight average is a centered Gaussian random variable with variance s^2. The equivalence with a two level MAP estimate is shown in appendix B.

With this modification, we obtain the cost C_{ANI} minimized by Anisotropic Noise Injection (ANI), defined as follows:

$$C_{\mathrm{ANI}}(f) = \frac{1}{\ell}\sum_{i=1}^{\ell}\left(\boldsymbol{w}^T\boldsymbol{x}_i + \theta - \boldsymbol{y}_i\right)^2 + \sum_{j=1}^{d}\sigma_j^2 {w_j}^2 \ , \tag{3}$$

$$\text{with } \frac{1}{d}\sum_{j=1}^{d}\frac{1}{\sigma_j^2} = \frac{1}{\sigma^2} \ , \tag{4}$$

where the optimized variables are $\boldsymbol{w}$, θ and $\{\sigma_j^2\}_{j=1}^d$, and where σ^2 is the second level hyper-parameter. It is fixed during the estimation of $\boldsymbol{w}$, θ and $\{\sigma_j^2\}_{j=1}^d$ and its value is set in a separate step by the estimation of generalization error.

Modified versions of ridge regression and NI can be equivalently used for minimizing (3) for a given matrix $\boldsymbol{\Sigma}$. For ridge regression, $\boldsymbol{\Sigma}$ would be adapted under constraint (4) by minimizing the full Lagrangian of the constrained optimization problem.

With NI, the penalizer in (3) is obtained with a noise covariance matrix $\boldsymbol{\Sigma}$, with $\boldsymbol{\Sigma}_{jj} = \sigma_j^2$. There are two ways of adapting the parameters of the noise distribution. The first solution would consist in adding some noise $\boldsymbol{\eta} \sim \mathcal{N}(0, \boldsymbol{I}_d)$, and multiply this noise by $\boldsymbol{\Sigma}^{1/2}$. The diagonal matrix $\boldsymbol{\Sigma}^{1/2}$ can then be optimized with constraint (4) by any kind of numerical optimization. This solution is likely to entail stability problems as σ_j can be very large. The second solution avoids this problem by using the parameters $1/\sigma_j$, which are bounded by constraint (4). It consists in multiplying the inputs by the gain matrix $\boldsymbol{\Sigma}^{-1/2}$ before injecting the noise $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$. The gain matrix is then optimized with constraint (4) by an iterative numerical optimization. After training, the gain matrix can be incorporated in the first layer of weights $\boldsymbol{W}_1$ by replacing $\boldsymbol{W}_1$ by $\boldsymbol{\Sigma}^{-1/2}\boldsymbol{W}_1$. The algorithm for a multi-layer perceptron with $H-1$ hidden layers is outlined in table I.

As for ridge regression, several values of σ^2 are tested, and the one with smallest estimate of generalization error is picked. Note that using $\boldsymbol{\Sigma}^{-1/2}$ does not require to compute the inverse or the square root of $\boldsymbol{\Sigma}$. So, reducing the dimension of the input space (instead of the number of variables) is easily obtained with a full matrix $\boldsymbol{\Sigma}^{-1/2}$. The analogous of constraint (4) is to constrain the norm of all the column vectors of $\boldsymbol{\Sigma}^{-1/2}$ to be $1/\sigma$.

IV. ESTIMATION OF RELEVANCES

A. LINEAR CASE

Equations (3) and (4) show that ANI derives d penalization parameters σ_j^2 from a single one σ^2, with $\sigma^2/d \leq \sigma_j^2 < \infty$. When σ_j^2 goes to infinity, variable j is discarded, in the opposite extreme, where $\sigma_j^2 = \sigma^2/d$, only variable j is used to estimate $\boldsymbol{y}$, thus σ/σ_j can be used as a relevance index for input variable j.

To understand how relevances can be assigned to each variable, it is convenient to think of ANI as a re-estimation scheme, alternating optimization with respect to $\boldsymbol{w}$ and $\{\sigma_j^2\}_{j=1}^d$. Let us consider the case where one input variable k is poorly correlated with the output. With $\sigma_j^2 = \sigma^2,\ j = 1, \ldots, d$, and standardized inputs, optimizing over $\boldsymbol{w}$ yields a solution with w_k^2 small. The optimization over $\{\sigma_j^2\}_{j=1}^d$ affects only the second term of C_{ANI}. The penalization are redistributed, with a tendency in increasing σ_k^2, as this allows to decrease $\sigma_j^2,\ j \neq k$. The change is not drastic, as constraint (4) allows only small decreases of $\sigma_j^2,\ j \neq k$ for a large increase of σ_k^2. Discarding one variable is thus only an extreme solution in the continuum of admissible ones. The process is iterated. In our scenario, optimizing over $\boldsymbol{w}$ with the new penalization should give another solution with smaller w_k^2, yielding a more stringent penalization of variable k, and iterations would go on until convergence.

This example illustrates three important features of ANI:

1. ANI rates relevances according to w_j^2. This rating is conventional, and the originality of the method lies in the re-estimation of σ_j^2, which implies re-estimation of w_j.
2. Inputs should be standardized to interpret σ/σ_j as the relevance for feature j. ANI, like NI is not scale invariant.
3. Constraint (4) states that a large increase in one penalization parameter allows only a small decrease among the other ones. The algorithm should thus be conservative, as strong evidence is needed for ANI to depart significantly from the initial solution where all inputs have the same relevance.
The last point is especially important, as for any adaptive scheme where relevances are estimated from data. ANI is expected to be less biased than standard NI when all inputs do not have the same relevance, but also more variable due to the additional degrees of freedom provided by optimizing over $\{\sigma_j\}_{j=1}^d$. Here, we can expect benefits from adaptativity as variability is limited by (4).

The analysis of the effective number of parameters, given in appendix C provides arguments supporting that the algorithm should be competitive compared to NI, and that it should be more stable than subset selection procedures. It results that suppressing a variable should not be carried out without sufficient support from data. For example, if two inputs are highly correlated, it is unlikely that they will be penalized differently. In this respect, ANI is very different from subset selection procedures.

B. NON-LINEAR CASE

For linear perceptrons, the interest of NI is rather limited since it is equivalent to weight-decay. For MLPs, as a randomization technique, NI is a means to escape from local minima. More important, it offers a control on the complexity of the solution in a way similar to weight decay. It is not simply equivalent to penalize weights from the input layer to the first hidden layer. The added noise is propagated throughout the network, yielding penalization on every layer, except for biases. The analytical form of this penalization is not known as the noise distribution at the outputs of hidden cells is complex, but the effects of NI on the network function have been studied and, at this level, NI is better understood than weight penalization techniques. Links have been drawn with Nadaraya-Watson kernel regression [13], Tikhonov regularization [5], and partial differential equations [6]. A formal comparison between NI and weight decay in the non-linear case is still to be done, but appendix A provides a discussion illustrating some differences between these two penalization schemes.

As in the linear case, adaptive noise injection sets the complexity of the network and the relevance of input variables by a single hyper-parameter σ^2. The set of relevances could be summarized to define the effective input dimension space, or the effective number of input variables. Selecting the numbers of parameters and variables by a single parameter is natural in linear regression, where there is one parameter per input variable. Conversely the number of inputs of an MLP does not define its complexity, and setting relevances and the effective number of parameters with a single by a single hyper-parameter is unusual.

However, minimizing C_{NI} (1) leads to have the Nadaraya-Watson smoother as target [13]. For this regressor, the effective number of parameters is controlled only by the width of the kernel. With ANI, the target is still a Nadaraya-Watson smoother, and, as described in annex C, σ^2 is still the only parameter controlling complexity. The double role of σ^2 limits the adaptativity of the method, but it is also what makes the algorithm easy to use, requiring a single hyper-parameter tuning.

The three main points stated for the linear case can be transcribed as follows:
1. In the non-linear case, ANI rates relevances by the shape of the kernel (neighborhood) used for smoothing, *i.e.* the shape of the noise distribution. Isotropic regression is performed in the metric defined by $\mathbf{\Sigma}$.
2. Inputs should still be standardized to interpret σ/σ_j as the relevance for feature j.
3. Constraint (4) states that the kernel used for smoothing is of minimum volume for $\sigma_j^2 = \sigma^2,\ j = 1, \ldots, d$, which should correspond to maximum complexity. The algorithm should thus be conservative.
As for the linear case, the last point is important regarding the stability of the algorithm.

V. IMPLEMENTATION

This section gives details on the implementation of NI and ANI. It is motivated by the will to ensure reproducible results. The reader familiar with the training of neural networks in the MAP framework may skip to section VI.

A. Optimization issues

The optimization of an MLP is a NP-hard problem [14]. Thus, even if ideally the optimization algorithm should not have any influence on the solution, it is indeed important. Here, batch learning with a variable step algorithm [15] is performed. Classical stopping criteria on convergence of training error, weight gradient or weights can not be used as is, since the algorithms are stochastic due to the injection of random noise. However, during training, there should be at first a decreasing trend in training error which should disappear at the equilibrium distribution. Here, optimization is stopped when the training error increases compared to its value 100 updates (epochs) ago.

As NI and ANI are randomization techniques, there are less sensitive to weights initialization. However, precautions should be taken to avoid inconsistencies due to optimization problems, such as a larger training error for a smaller value of the penalization parameter for a given training set. Here, training is performed sequentially, starting with the largest value of σ^2. For this value, a single training is performed with weights independently drawn from a centered Gaussian distribution of variance 10^{-2}. Then, the final weights for the current value of σ^2 are used to initialize the network for the next trial value. Thus only one training is performed per training set and value of σ^2.

B. Hyper-parameter trial values

The MAP classical paradigm requires two steps for setting the hyper-parameter. First a range of values is considered, then, among several trial values in this range, the one minimizing an estimate of generalization error is selected.

A rapid choice of the range of values is performed by splitting the training set into two parts to get a training and a validation set, the error on the validation set being the estimate of generalization error. Starting with σ^2 to be of the order of the mean input variances, the trial values are obtained subsequently by dividing the previous one by ten, until overtraining is met. Overtraining is recognized by a significant increase of the (estimate of) generalization error after its minimum has been reached.

With this rough scheme, one obtains an interval, centered on the minimum of the estimated generalization error, where significant under and overfitting occur. Here, we used five to seven equidistant trial values in this interval *in log scale*.

C. Hyper-parameter selection

Using a validation set for estimating σ^2 implies that one part of the original data set (the validation set) is not used to estimate the network weights. The value of σ^2 minimizing the estimate of generalization error on the validation set could be used on the whole data set. However, the optimal value of σ^2 depends on the sample size. The larger the sample size is, the smaller the optimal of σ^2 gets. Resampling procedures, such as cross-validation and bootstrap make an extensive use of the training set, using the whole set for estimating the network weights, while σ^2 is estimated (for the right sample size) with perturbated versions of the training set. Here, the optimal value of σ^2 is estimated by bootstrap.

Once a set of candidates values for σ^2 is selected, the training is performed on the whole data set. A sequence of models indexed by σ^2 is thus obtained, from which one has to be chosen. This selection is performed by bootstrap estimates of generalization error. For each net size, $B = 20$ balanced bootstrap replicates of the training sample are computed. Balanced bootstrap ensures that each training example appears exactly B times over all bootstrapped sets. There are several bootstrap

estimates of generalization error. The .632 bootstrap estimate, advocated by Efron and Tibshirani [16] was used here:

$$\begin{aligned} E_{\text{boot}}(\sigma^2) &= .368\frac{1}{\ell}\sum_{i=1}^{\ell} l(f(\boldsymbol{x}_i;\sigma^2), y_i) \\ &\quad +.632\frac{1}{\ell}\sum_{i=1}^{\ell}\frac{1}{B_i}\sum_{b\in\mathcal{B}_i} l(f_b(\boldsymbol{x}_i;\sigma^2), y_i) \ , \end{aligned} \tag{5}$$

where l is the loss function, $f(\cdot;\sigma^2)$ is the network function estimated from the whole training sample for hyper-parameter σ^2, $f_b(\cdot;\sigma^2)$ is estimated from the b^{th} bootstrap sample, $\mathcal{B}_i$ is the set of indices of the bootstrap samples not containing the i^{th} data point, and B_i is the number of such bootstrap samples.

For six trial values for σ^2, estimating the neural network weights requires thus $(20+1)\times 6 = 126$ trainings. This might represent too much computational effort for large training sample, but for these problems, it should be possible to use a validation set, thus reducing considerably the number of trainings.

VI. COMPARISONS WITH NI AND ARD

In this section, Anisotropic Noise Injection is compared to Noise Injection and to Automatic Relevance Determination on two artificial datasets. The results for ARD are borrowed from Neal [10]. In order to allow comparison, the neural net architectures used by Neal are also used here.

A. REGRESSION

The task of the robot arm problem is to estimate the continuous mapping from joint angles to position for a robot arm model. There are two real input variables for the joint angles, and two real output variables for the position. The model is as follows:

$$y_1 = 2\,\cos(x_1) + 1.3\,\cos(x_1+x_2) + \varepsilon_1 \tag{6}$$

$$y_2 = 2\,\sin(x_1) + 1.3\,\sin(x_1+x_2) + \varepsilon_2 \ , \tag{7}$$

where ε_1 and ε_2 are independent Gaussian noise of standard deviation 0.05, x_1 is drawn uniformly in $[-1.932, -0.453]\cup[0.453, 1.932]$, and x_2 is drawn uniformly in $[0.534, 3.142]$. Figure 1 shows the robot arm model in an admissible position.

In this version four input variables are added:

$$x_3 = x_1 + 0.02\,\varepsilon_3 \quad x_5 = \varepsilon_5 \tag{8}$$

$$x_4 = x_2 + 0.02\,\varepsilon_4 \quad x_6 = \varepsilon_6 \ , \tag{9}$$

where ε_3, ε_4, ε_5, and ε_6 are independently drawn from $\mathcal{N}(0,1)$. We thus have a regression problem with two output variables and six input variables, where x_5 and x_6 are irrelevant to predict y_1 and y_2. In the absence of x_1 and x_2, x_3 and x_4 would be useful for estimating the outputs, but in the present design, they should ideally be discarded.

The training set contains $\ell = 200$ examples. The same training examples are used for NI and ANI. The performances are evaluated on a single independent test set of 10000 examples, and the experiment is repeated for ten independent training sets. As in [10], the network has a single hidden layer of 16 hyperbolic tangent units and linear output units, *i.e.* with 146 parameters, and training is performed with the mean least squares loss function.

The results regarding mean squared error are reported on table II. For a test set size of 10000, the standard error on these results for each training set is about $6\ 10^{-5}$.

The average errors for NI and ANI are respectively 12.72 10^{-3} and 8.95 10^{-3}. Averaging over training set gives the ranking Bayesian learning with ARD, Bayesian learning without ARD, ANI and NI. The minimum reachable error is the noise variance, *i.e.* 5 10^{-3}, so that the Bayesian methods are very close to the true regression function.

Figure 2 displays the boxplot of relevance indices σ_j/σ for the six input variables. It shows that the improvement of ANI over NI is mainly explained by the down-weighting of the two last irrelevant variables. ANI is not successful in discarding x_3 and x_4. The test errors obtained by NI and ANI are much higher than the ones of Bayesian learning, but ANI is still better (in average and for eight out of ten training sets) than a prediction made by test points, for which the error would be twice the noise variance, *i.e.* 10^{-2}. Thus, although the moderate sample size, the ANI estimate of the regression function is closer to the true regression function than test points.

The noise variance σ^2 (4) was selected by .632 bootstrap from six values uniformly spaced on a log scale from the ranges $[10^{-5}, 10^{-1}]$ for NI, and $[10^{-6}, 10^{-1}]$ for ANI. The average estimated optimal values for σ^2 are about 7 10^{-3} for NI and 5 10^{-4} for ANI. Estimating σ^2 by .632 bootstrap is not responsible for the relative failure of NI and ANI, as it induces a relatively small loss compared to the smallest test error observed for each training set: in average 1.48 10^{-3} for NI and 0.65 10^{-3} for ANI. The poorest performances are likely to be mainly due to the badly conditioned optimization problem. The correlations between x_1 and x_2 on one side, and x_3 and x_4 on the other side are very high (about $1 - 10^{-4}$). The condition number is improved with NI, which turns the input covariance matrix $(\boldsymbol{X}^T\boldsymbol{X})$ to $(\boldsymbol{X}^T\boldsymbol{X} + \sigma^2\boldsymbol{I}_d)$, but the estimated optimal values for σ^2 are too low to provide a significant improvement. More than 10 000 epochs were sometimes required to meet our crude stopping criteria for the estimated optimal values of σ^2.

This problem is illustrated by figure 3, which shows the contour plot of

$$\begin{aligned} C(\alpha,\beta) = \frac{1}{\ell} \sum_{i=1}^{\ell} & (y_{1i} - 2 * \cos(\alpha x_{1i} + \beta x_{3i}) - \\ & 1.3 * \cos(\alpha x_{1i} + \beta x_{3i} + x_{2i}))^2 + \\ & (y_{2i} - 2 * \sin(\alpha x_{1i} + \beta x_{3i}) - \\ & 1.3 * \sin(\alpha x_{1i} + \beta x_{3i} + x_{2i}))^2 \end{aligned} \tag{10}$$

for the fifth learning set, which is typical regarding performances of NI and ANI.

The minimum cost is reached for $(\alpha, \beta) = (1.03, -0.03)$ ($C = 3.96\ 10^{-3}$), very close to the true values $(\alpha, \beta) = (1, 0)$ but the nearly degenerated shape of the elliptic isolevels, with principal axis on $\alpha + \beta = 1$ shows the optimization difficulties of in this extremely simplified problem. Here, the cost is higher for $(\alpha, \beta) = (1.04, -0.02)$ ($C = 5.20\ 10^{-3}$) than for $(\alpha, \beta) = (0.31, 0.69)$ ($C = 4.98\ 10^{-3}$). Our stochastic algorithm is thus likely to stop far from the optimal solution. Neal reports that in his experiments, 16 hours of computation time were required to train a Bayesian model.

B. DISCRIMINATION

The noisy LED display was originally used to evaluate classification trees [17]. The task is to estimate which digit is indicated on a standard seven-segment LED display, where each segment is wrong with probability 0.1. Each segment is a binary input variable numerically coded as +0.5/-0.5. Figure 4 shows the ten digits, together with the feature number used for coding.

In the version of the problem considered here, seventeen additional irrelevant variables are added, taking the values +0.5/-0.5 with equal probability. There are ten binary outputs corresponding to the ten digits.

The training set contains 20 examples of the 10 digits, *i.e.* $\ell = 200$ examples. As in the previous section, the same training examples are used for NI and ANI. The performances are evaluated on

a single independent test set of 5000 examples, and the experiment is repeated for ten independent training sets.

As in [10], the neural net contains one hidden layer of eight hidden hyperbolic tangent units, plus direct connections from inputs to outputs (404 parameters). The network output units have a softmax function [18], and training is performed with the cross-entropy loss function:

$$l(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}) = -\sum_{k=1}^{K} y_k \ln(f_k(\boldsymbol{x})) \quad , \tag{11}$$

where K is the number of classes, and y_k and f_k are respectively the k^{th} target value, and the k^{th} output of the network.

The results regarding classification error rates are reported on table III. For a test set size of 5000, the standard error on each classification error rate is about 0.6%, so that differences of 2% on the same training set are significant at a 5% level [10].

The differences between ANI and NI are thus significant, as are the differences between Bayesian learning with and without ARD (except for the third set). The average error rates for the Bayesian neural net models on the three training sets are 35.9% without ARD, and 31.7% with ARD. The rates on ten training sets are 33.1% for NI, and 27.6% for ANI. Neal reports that the optimal classification rule (based on knowledge of misclassification rate. For all training sets used here, the solutions provided with ANI are thus nearly optimal.

The ranking of the four methods is as follows: ANI, Bayesian learning with ARD, NI, and Bayesian learning without ARD. The significance of differences between the two groups of method is questionable, as learning was not performed on the same training sets. However, the standard deviation of the average error rates for ANI and NI are respectively 0.7%, and 1.7%, and support the ranking provided above.

With more than two third of irrelevant features, the improvement of ANI over NI was expected, and is explained by the input weight magnitudes, as shown on figure 5. These plots show the square roots of the average squared magnitude of fan-out weights for the twenty-four input units (the seven relevant ones followed by the seventeen irrelevant ones).

Although NI already spots relevant features, ANI goes far beyond by assigning negligible weights to the irrelevant inputs. Note also that the third (relevant) feature is sometimes heavily penalized by ANI. This is due to the fact that, for correct patterns, the third LED is always on, except for digit 2 (see figure 4).

The improvement of NI over the more refined Bayesian procedures is surprising, but might be explained by a slightly inadequate choice of priors for these models, as discussed by Neal [10, p. 116–122]. This is confirmed by the comparison of figure 5 with the equivalent figure supplied in [10]. Indeed, plots of the posterior distributions of the same variables in the ARD model shows that, with the priors used by Neal, the ARD model is much less selective than ANI. The ratio between relevant and irrelevant features is about 10, *i.e.* intermediate between NI and ANI as for its ranking regarding classification error rates.

The noise variance σ^2 (4) was selected by .632 bootstrap from six values uniformly spaced on a log scale in the ranges $[10^{-2}, 10^{1}]$ for both NI and ANI. This estimation was quite efficient, with an average loss over the true optimal performance of 2% for NI and 0.5% for ANI. The average estimated optimal values for σ^2 are about 10^{-1} for NI and 4 10^{-1} for ANI, with little variations about these values. For ANI, the average input noise variances σ_j^2 are about 2 10^{-2} for relevant features, and 15 for irrelevant ones. Using the same algorithm for binary and real inputs might have seem inappropriate, but the Gaussian noise injected on binary features is equivalent to a Bernoulli perturbation. Indeed, if the boundary between on and off values is between -0.5 and +0.5 for each feature, the noise variances given above correspond to 5% of status flip for each feature with NI, virtually no change for relevant features, and 45% for irrelevant ones in ANI.

VII. FURTHER EXPERIMENTS

In this section, we give experimental results illustrating the capacity of the proposed algorithm to indicate gradual relevance of input variables while solving a learning task. We borrowed simulated data and results from Friedman [19]. These data were created to test flexible (thus adaptive) metric nearest neighbor classification. These tests are well adapted to our algorithm in the sense that it is also based on an adaptive metric. On the other hand, the nearest neighbor metrics are local, whereas we use a global metric. Regarding generalization error, the examples are rather favorable to global metrics, but Friedman provides experimental results obtained by CART [17], which is also global method. We briefly summarize the description of the examples below.

Example 1: There are $d = 10$ input variables, $\ell = 200$ training observations equally divided in two classes. The inputs are generated from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ for the first class, and $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$ for the second class, with $\{m_j = \sqrt{j}/2\}_{j=1}^d$ and $\boldsymbol{C} = \text{diag}\{1/\sqrt{j}\}_{j=1}^d$.

Example 2: Same as above except that $\{m_j = \sqrt{d-j+1}/2\}_{j=1}^d$.

Example 3: There are $d = 10$ input variables, $\ell = 200$ training observations equally divided in two classes. The inputs are generated from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ for the two classes. The classes are defined by: $\sum_{j=1}^d x_j^2/j \leq 2.5 \Rightarrow$ class 1, and class 2 otherwise.

Example 4: There are $d = 10$ input variables, $\ell = 500$ training observations and two classes. The inputs are generated from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ for the two classes. The classes are defined by: $\sum_{j=1}^d x_j^2 \leq 9.8 \Rightarrow$ class 1, and class 2 otherwise.

Example 5: Same as example 3, except that the two classes are defined by: $\sum_{j=1}^d x_j \leq 0 \Rightarrow$ class 1, and class 2 otherwise.

Example 6: This is the classical waveform example [17]. There are $d = 21$ input variables, $\ell = 300$ training observations equally divided in three classes. Each point $\boldsymbol{x}_i$ is a random combination of two out of three basic waveform, with noise added: class 1 $x_{ij} = u_i h_1(j) + (1-u_i)h_2(j) + \varepsilon_{ij}$; class 2 $x_{ij} = u_i h_1(j) + (1-u_i)h_3(j) + \varepsilon_{ij}$; class 3 $x_{ij} = u_i h_2(j) + (1-u_i)h_3(j) + \varepsilon_{ij}$, with $h_1(j) = (6 - |j-7|)_+$, $h_2(j) = (6 - |j-15|)_+$, $h_3(j) = (6 - |j-11|)_+$, u_i are generated from a uniform distribution on [0, 1], and ε_{ij} from a standard Gaussian distribution.

ANI, like NI is not scale invariant, so that the input variables should be normalized to produce sensible estimates of the decision rule and relevances. The normalization advocated by Friedman [19] is used here. It is based on a robust scale measure, *i.e.* the interquartile range:

$$\text{scale}_j = \left[x_{(3\ell/4)j} - x_{(\ell/4)j}\right]^{-1} \quad , \tag{12}$$

where $x_{(3\ell/4)j}$ and $x_{(\ell/4)j}$ are respectively the upper and lower quartile of the j^{th} input variable. Once the inputs are multiplied by diag(scale), ANI returns the relevance indices of the normalized inputs.

For all examples, ten independent training samples of size ℓ are generated, and a multi-layer perceptron with a single hidden layer of 10 units is trained with ANI. The number of parameters of the network (including biases) is $(d+1) \times 10 + (10+1) \times K$, where K is the number of classes. The networks are trained with the cross-entropy loss function (11), and σ^2 is estimated by .632 bootstrap.

Friedman's results were computed on 10 independent generalization sets of size 2000. ANI test error rates are estimated on one independent generalization set of size 5000. The standard deviation of estimated test error over the 10 training samples is also reported in Table IV. The imprecision due to the finite size of the generalization test is not taken into account, as it is dominated by the variability over training samples.

Our purpose here is mainly to illustrate the capabilities of ANI to allocate relevances to input variables. It would not make much sense if the generalization performances were bad, but table IV shows that ANI performs well compared to the other algorithms, as the classification error is either the lowest, or close to the lowest one, except for example 3. Note that for the latter, the poor performance is mainly due to the loss incurred by estimating σ^2 by .632 bootstrap, which is much higher than

for the other examples (4.7% compared to 1.9% for example 4 and less than 0.4% for all the other examples).

The estimated relevance indices are illustrated on figure 6 by boxplots summarizing the results on the ten training samples. The relevance indices (σ_j/σ) are normalized, so that they would be equal to one with NI (equal relevances).

The boxplot for example 1 shows that the variables with higher coordinates, where classes are more separated, are more relevant. For example 2, all but higher coordinates are about equally relevant. For this problem, the error rates based on one input variable and the knowledge of the true distribution are between 21% and 25% for the seven first coordinates, and between 27% and 33% for the three last ones. For example 3, the smaller coordinates are rated more relevant. A pattern can be correctly classified when a large value of x_1^2 is observed, but the value of x_{10}^2 is never discriminant by itself. Note that for this example, the variability of relevances for the more important features is partly explained by the bad estimation of σ^2. For examples 4 and 5, all variables have the same relevance, and the ratings do not vary much, especially for example 5, for which multi-layer perceptrons could be perfect discriminators. Finally, in example 6, the three waveforms defining the classes (respectively centered in 7, 11 and 15) are visible on the boxplot. The classes being convex combination of two waveforms, small values of x_1 and x_{21} are equally likely for the three classes, but large values of x_7, x_{11} or x_{15} are unlikely for one out of the three classes (respectively class three, one and two).

These experiments show that the average gradual relevances assigned by ANI are meaningful. When all inputs are equally relevant, this is faithfully reflected by the relevance indices. Variability is lower for more relevant features. Less relevant inputs differ in number and coordinates according to the training samples. The distribution of relevances is thus bimodal, which results in higher variability. Note that for a given training sample, this is often reflected on the distribution of relevances on the bootstrap samples.

VIII. DISCUSSION

This paper presented how NI can be modified in order to determine relevant inputs. Rather than suppressing variable, penalties are distributed on input features. The proposed algorithm retains the simplicity and ease of use of NI. For each training, the additional computational cost is limited to the optimization of the noise variances $\{\sigma_j^2\}_{j=1}^d$, where d is the number of inputs. There is still a single hyper-parameter to tune, so that the number of training required by both algorithm is identical. Regarding this point, the randomization due to the added noise is also beneficial, at it provides a means to escape from local minima for ANI as for NI. All experimental results given in this paper were obtained from a single starting point.

The simplicity and ease of use of the algorithm are due to the double role of the hyper-parameter σ^2. It is a scale parameter in the input feature space, setting neighborhood sizes, and a complexity parameter tuning the effective number of parameters. This double role is similar to the one of kernel width in Nadaraya-Watson kernel regression. On the one hand it limits the adaptativity of the algorithm, since locality and complexity are tied, on the other hand it avoids the use of multiple hyper-parameter.

From a Bayesian point of view, ANI is shown to be connected to maximum *a posteriori* estimation with hierarchical priors. The form of the prior distributions is given for linear regression without intercept (linear perceptron without bias unit). Furthermore, for this particular case, ANI is shown to be equivalent to standard (1-level) maximum *a posteriori* estimation with Laplace priors.

The bounds on the effective number of parameters, provided in the linear and non-linear cases, support that the stability of this adaptive scheme is enforced by the constraints applied to the admissible set of relevance indices. The elaborated arguments are not complete, since the computation of the effective number of parameters is not tractable for adaptive schemes, but they are back up by experimental results.

Experimentally, stability is shown by using several training sets generated from the same distribution.

Generalization error variability is lower for ANI than for NI, and the estimated relevances on the different training sets are similar. Examples with equally relevant features show also that the algorithm is conservative, *i.e.* features are not discarded or heavily penalized without due cause. The experiments also demonstrate the improvement that can be brought up by ANI upon NI, and that ANI can be competitive with the more sophisticated automatic relevance determination model developed by MacKay and Neal [10].

Acknowledgments

The author would like to thank S.Canu and M. Masson for fruitful discussions and comments on earlier drafts of this paper.

References

[1] L. Breiman, "Heuristics of instability and stabilization in model selection," *The Annals of Statistics*, vol. 24, no. 6, pp. 2350–2383, 1996.

[2] T.K. Leen, "From data distribution to regularization in invariant learning," *Neural Computation*, vol. 7, no. 5, pp. 974–981, 1995.

[3] A.R. Webb, "Functional approximation by feed-forward networks: A least-squares approach to generalization," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 363–371, 1994.

[4] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural Computation*, vol. 8, no. 3, pp. 643–674, Apr. 1996.

[5] C.M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, Jan. 1995.

[6] Y. Grandvalet, S. Canu, and S. Boucheron, "Noise injection: theoretical prospects," *Neural Computation*, vol. 9, no. 5, pp. 1093–1108, 1997.

[7] J. Sietsma and R.J.F. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, 1991.

[8] K. Matsuoka, "Noise injection into inputs in back-propagation learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 436–440, 1992.

[9] Y. Grandvalet and S. Canu, "A comment on noise injection into inputs in back-propagation learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 4, pp. 678–81, Apr. 1995.

[10] R. M. Neal, *Bayesian Learning for Neural Networks*, Number 118 in Lecture Notes in Statistics. Springer, New York, 1996.

[11] J. Larsen, C. Svarer, L.N. Andersen, and L.K. Hansen, "Adaptive regularization in neural network modeling," in *Neural Networks: Tricks of the Trade*, G.B. Orr and K. Mller, Eds. 1998, vol. 1524 of *Lecture Notes in Computer Science*, pp. 113–132, Springer.

[12] M.J.L. Orr, "Local smoothing of radial basis function networks," in *International Symposium on Artificial Neural Networks, Hsinchu, Taiwan*, 1995.

[13] L. Holmström and P. Koistinen, "Using additive noise in back-propagation training," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 24–38, Jan. 1992.

[14] E. Amaldi, "On the complexity of training perceptrons," in *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. North-Holland, Amsterdam, 1991.

[15] R.W. Harris, D.M. Chabries, and A. Bishop, "A variable step (VS) adaptive filter algorithm.," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 2, pp. 309–316, Apr. 1986.

[16] B. Efron and R.J. Tibshirani, *An Introduction to the Bootstrap*, vol. 57 of *Monographs on Statistics and Applied Probability*, Chapman & Hall, New York, 1993.

[17] L. Breiman, J.H. Friedman, R. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA., 1984.

[18] J.S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, relationships to statistical pattern recognition," in *Neuro-computing: Algorithms, Architectures and Applications*, F. Fogelman Soulié and J. Hérault, Eds. 1990, pp. 227–236, Springer.

[19] J.H. Friedman, "Flexible metric nearest neighbor classification," Tech. Rep., Stanford University, Stanford, CA., Nov. 1994.

[20] D.J.C. MacKay, "A practical Bayesian framework for backprop networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.

[21] J. Moody, "Prediction risk and architecture selection for neural networks," in *From statistics to neural networks, theory and pattern recognition applications*, V. Cherkassky, J.H. Friedman, and H. Wechsler, Eds. 1994, vol. 136 of *NATO ASI series F: Computer and systems sciences*, pp. 147–165, Springer.

[22] R.J. Tibshirani and K. Knight, "The covariance inflation criterion for adaptive model selection," *Journal of the Royal Statistical Society, B*, vol. 61, no. 3, pp. 529–546, 1999.

[23] T.J. Hastie and R.J. Tibshirani, *Generalized Additive Models*, vol. 43 of *Monographs on Statistics and Applied Probability*, Chapman & Hall, New York, 1990.

Appendix

I. NI *versus* weight decay

The understanding of weight decay regarding the network function is too limited to allow a formal comparison with NI. However, for networks with an infinite hidden layer, Neal showed that Bayesian learning with a Gaussian prior converges to a Gaussian process [10, chapter 2]. This result hints a connection of weight-decay with kernel regression which underlines the double role of the penalization parameter:
1. when applied to the input-to-hidden weights, it is similar to a bandwidth parameter since it sets the slope of the sigmoid functions ;
2. when applied to the hidden-to-output weights, it is similar to the regularization parameter of kernel methods which controls the bias/variance tradeoff.
These two roles justify the use of two different penalization parameters suggested by MacKay [20].

For NI, the variance of the injected noise acts as a penalization parameter, and it is similar to a bandwidth parameter. NI is a stochastic local averaging technique, hence it has no second parameter controlling the tightness of the fit. This simple handling of complexity with a single hyper-parameter motivated our interest in NI.

NI can be expected to be more effective than weight decay when the double role of its penalization parameter requires a compromise. Compared to the more sophisticated technique with two penalization parameters, NI is easier to tune, often competitive, and sometimes more appropriate.

This is illustrated on the following experiment: there is one input variable uniformly distributed on $[-1, 1]$, and one output variable $y = H(x) + \varepsilon$, where H is the Heaviside function and ε is normally distributed with mean zero and variance 10^{-2}. A training sample consists of 10 examples. To exhibit the differences between the two complexity control schemes, an oversized network, with 10 hidden units is trained with the quadratic loss. The experiment is repeated for 100 training samples to provide an estimate of the bias and variability for weight decay (two penalization parameters) and for NI. Figure 7 displays the average of the 100 networks responses for both methods optimally tuned regarding mean prediction error.

The optimal estimates of NI and weight decay have approximately the same variance (integrated variance is about $3.7\ 10^{-1}$), but NI is significantly less biased (integrated square bias of $2.7\ 10^{-1}$ vs. $3.7\ 10^{-1}$). Weight decay imposes low magnitude weights, which provide smoothness on the whole input domain. All abrupt changes are penalized, and the network function has the "bumpy" behavior which characterizes generalized spline methods. The local invariance required by NI results also in a smooth network function, but with a trend towards locally constant functions, which is more appropriate in this setting.

II. MAP estimation

Anisotropic Noise Injection applied to a linear perceptron without bias term can be interpreted as a two level MAP estimation scheme. The proof is given here and with a single output unit to lighten notation.

In the two level ARD scheme described by Neal [10], the network weights are given Gaussian priors $\mathcal{N}(0, s_j^2)$, and an inverted Gamma distribution is used for setting the distribution of the hyper-parameters $\{s_j^2\}_{j=1}^d$. This choice is motivated by mathematical convenience. Here, $\{s_j^2\}_{j=1}^d$ are given exponential priors. The prior on $\tau_j = s_j^2$ is:

$$P(\tau_j) = \lambda \exp(-\lambda \tau_j) \ ,$$

where λ is the second level hyper-parameter. The prior on s_j is thus a Weibull distribution:

$$P(s_j) = 2\lambda s_j \exp(-\lambda s_j^2) \ . \tag{13}$$

The data points $(\boldsymbol{x}_i, y_i)$ of the learning set $\mathcal{L}$ are assumed to be independently drawn from some distribution such that $(y|\boldsymbol{x})$ is Gaussian: $P(y|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}^T\boldsymbol{w}^*, \sigma_y)$. The MAP estimate maximizes Post $= P(\boldsymbol{w}, \{s_j\}_{j=1}^d|\mathcal{L})$:

$$\begin{aligned}
\text{Post} &\propto P(\mathcal{L}|\boldsymbol{w}, \{s_j\}_{j=1}^d) P(\boldsymbol{w}|\{s_j\}_{j=1}^d) P(\{s_j\}_{j=1}^d) \\
&\propto \prod_{i=1}^{\ell} \frac{1}{\sqrt{2\pi}\sigma_y} \exp\Big(-\frac{(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2}{2\sigma_y^2}\Big) \\
&\quad \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}s_j} \exp\Big(-\frac{w_j^2}{2s_j^2}\Big) \prod_{j=1}^{d} 2\lambda s_j \exp(-\lambda s_j^2) \\
&\propto \Big(\frac{1}{\sqrt{2\pi}\sigma_y}\Big)^{\ell} \exp\Big(-\frac{1}{2\sigma_y^2}\sum_{i=1}^{\ell}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2\Big) \\
&\quad \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}s_j} \exp\Big(-\sum_{j=1}^{d}\frac{w_j^2}{2s_j^2}\Big) \\
&\quad 2^d\lambda^d \prod_{j=1}^{d} s_j \exp\Big(-\lambda\sum_{j=1}^{d} s_j^2\Big) \ .
\end{aligned}$$

The MAP estimate equivalently minimizes $L = -\log(P(\boldsymbol{w}, \{s_j\}_{j=1}^d|\mathcal{L}))$:

$$\begin{aligned}
L &= \frac{1}{2\sigma_y^2}\sum_{i=1}^{\ell}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \sum_{j=1}^{d}\log(s_j) \\
&\quad + \sum_{j=1}^{d}\frac{w_j^2}{2s_j^2} - \sum_{j=1}^{d}\log(s_j) + \lambda\sum_{j=1}^{d} s_j^2 + c \\
&= \frac{1}{2\sigma_y^2}\sum_{i=1}^{\ell}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \frac{1}{2}\sum_{j=1}^{d}\frac{w_j^2}{s_j^2} + \lambda\sum_{j=1}^{d} s_j^2 + c \\
&\propto \frac{1}{\ell}\sum_{i=1}^{\ell}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \frac{\sigma_y^2}{\ell}\sum_{j=1}^{d}\frac{w_j^2}{s_j^2} + \frac{2\lambda\sigma_y^2}{\ell}\sum_{j=1}^{d} s_j^2 + c' \ , \qquad (14)
\end{aligned}$$

which is, up to the irrelevant constant c', recognized as the Lagrangian of C_{ANI} (3) under constraints (4), provided that $s_j^2 = \sigma_y^2/\ell\sigma_j^2$, and λ is proportional to the Lagrangian multiplier corresponding to constraint (4).

Note that if there is only one weight w_j per hyper-parameter s_j, the two-level MAP scheme with Gaussian first-level priors and Weibull second-level priors (13) is equivalent to a one-level MAP scheme with independent Laplace priors. Indeed, the distribution on parameter w_j is:

$$\begin{aligned}
P(w_j) &= \int_0^{\infty} P(w_j|s_j)P(s_j)\ ds_j \\
&= \int_0^{\infty} \frac{1}{\sqrt{2\pi}s_j} \exp -\frac{w_j^2}{2s_j^2}\ 2\lambda\, s_j \exp -(\lambda s_j^2)\ ds_j \\
&= \frac{\sqrt{2}\lambda}{\sqrt{\pi}} \exp\Big(-\sqrt{2\lambda}|w_j|\Big) \int_0^{\infty} \exp -\left(\frac{|w_j|}{\sqrt{2}s_j} - \sqrt{\lambda}s_j\right)^2 ds_j \\
&= \frac{\lambda}{\sqrt{2}} \exp\Big(-\sqrt{2\lambda}|w_j|\Big) \qquad (15)
\end{aligned}$$

Compared to the Gaussian distribution, the Laplace distribution puts slightly more mass near zero and in the tails. Compared to NI, ANI should thus have a tendency to produce solutions with smaller

and larger weights. This result is easily generalized when several weights are attached to one hyper-parameter, but the corresponding distribution is more complex.

III. Effective number of parameters

One of the goal of extending NI is to improve its performances when some inputs are more relevant than others. On the other hand, being more flexible should also induces a loss compared to the standard method when all inputs are equally relevant. The study of effective parameters provides some arguments supporting these expectations, and allows to compare the loss brought about by ANI and subset selection in the case most favorable to NI.

The effective number of parameters p_{eff} of an estimator [21] is defined by analogy with the number of parameters of the ordinary least squares linear model. Under some assumptions on the model correctness and data distribution [22], p_{eff} links the expected generalization error with the expected training error:

$$\mathbb{E}[E_{\text{gen}}] = \mathbb{E}[E_{\text{train}}] + \frac{2\sigma_y^2}{\ell} p_{\text{eff}} \ , \tag{16}$$

where E_{gen} and E_{train} are respectively the generalization error and the training error, σ_y^2 is the variance of $(y|\boldsymbol{x})$, and ℓ is the training sample size. Hence, $-2\sigma_y^2 p_{\text{eff}}/\ell$ is the bias of the training error used as an estimator of the generalization error.

For linear smoothers, p_{eff} can be analytically computed. A linear smoother is such that the fitted values at $\boldsymbol{x}_i,\ i = 1,\ldots,\ell$ are a linear combination of the observed output values $y_i,\ i = 1,\ldots,\ell$. Examples of these smoothers include ordinary linear regression, ridge regression Nadaraya-Watson kernel regression and smoothing splines [23]. Let $\boldsymbol{y} = (y_1,\ldots,y_\ell)^T$ be the vector of concatenated output values, and $\mathbf{f} = (\mathrm{f}(\boldsymbol{x}_1)\ldots\mathrm{f}(\boldsymbol{x}_\ell))^{\mathrm{T}}$ be the vector of concatenated fitted values. The effective number of parameters p_{eff} is the trace of the hat or smoothing matrix $\boldsymbol{S}$, such that $\mathbf{f} = \boldsymbol{S}\boldsymbol{y}$ [23].

A. Linear perceptron

For a given value of $\boldsymbol{\Sigma}$, the effective number of parameters of the minimizer of (3) is:

$$p_{\text{eff}} = \mathrm{tr}(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Sigma})^{-1}\boldsymbol{X}^T) \ , \tag{17}$$

with $\boldsymbol{X} = (\boldsymbol{x}_1\ldots\boldsymbol{x}_\ell)^T$.

In the orthogonal design where the inputs are uncorrelated and standardized ($\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_d$), then $p_{\text{eff}} = \sum_{j=1}^d 1/(1+\sigma_j^2)$. The set of σ_j maximizing p_{eff} under constraint (4) is the minimizer of:

$$-\sum_{j=1}^d \frac{1}{1+\sigma_j^2} + \lambda \sum_{j=1}^d \frac{1}{\sigma_j^2} \ ,$$

where λ is the Lagrange multiplier corresponding to constraint (4). The solution to this problem is $\sigma_j^2 = \sigma^2,\ j = 1\ldots d$. The choice of $\boldsymbol{\Sigma}$ maximizing the effective number of parameters is the one that gives the same relevance index to all variables, *i.e* the solution provided by NI. For all other choices of $\boldsymbol{\Sigma}$ considered by ANI, the training error is a less biased estimate of generalization error.

It is easily shown that optimizing C_{ANI} over $\{\sigma_j^2\}_{j=1}^d$ ensures that, for a given value of σ^2, ANI training error is lower than NI training error. It seems thus that ANI can only improve over NI, as E_{train} and p_{eff} (17) are both lower in ANI than in NI There is however a caveat, since p_{eff} (17) is the effective number of parameters for $\boldsymbol{\Sigma}$ fixed. It does not account for the effects of adapting $\boldsymbol{\Sigma}$ during learning. For ANI, p_{eff} (17) is thus likely to be a down-biased estimate of the effective number of parameters as defined by (16). Tibshirani and Knight [22] proposed to apply a correction computed by resampling methods, but no general analytical expression is available today.

In terms of generalization error, ANI can be worse than NI only if the benefits of minimizing jointly E_{train} and p_{eff} (17) are overcome by the variability of the solution due to the estimation of $\{\sigma_j^2\}_{j=1}^d$.

For example, one indeed expect worse performances for small training sets where all inputs have the same relevance, as the expected value of p_{eff} (17) for ANI should be close to the effective number of parameters for NI, and that the estimation of $\{\sigma_j^2\}_{j=1}^d$ should increase p_{eff} as defined by (16).

However, one also expect less variability with ANI than with more radical procedures like subset selection. First, note that in subset selection the variations of the smoothing matrix $\boldsymbol{S}$ are discontinuous and cover a wider range than in ANI. In the orthogonal design, these variations affect the eigenvalues of $\boldsymbol{S}$, which take either value 0 or 1 for subset selection, and are in the range $[0,\ d/(d+\sigma^2)]$ for ANI. The expectation of eigenvalues over training sets in ANI is within the range of $[0,\ d/(d+\sigma^2)]$. Furthermore, this convex combination of solution is itself an admissible solution of the algorithm. For subset selection, the expectation of eigenvalues over training sets is the probability of selecting each input variable, which is different from 0 or 1 as long as there is some variability in the choice of selected inputs, and is thus not reachable by the algorithm. Second, as departing from $\sigma_j^2 = \sigma^2$ induces a decrease of p_{eff} (17), it has to be supported by data (more stringent constraints should be compatible with the decrease of E_{train}). This support can only be given to a limited extent by the random fluctuations due to sampling. From these two points, it results that the fluctuations of ANI due to the estimation of $\{\sigma_j^2\}_{j=1}^d$ will be lower than the ones due to the choice of variables in subset selection.

B. Multi-layer perceptron

First, let us recall that for the quadratic or for the cross-entropy loss, C_{NI} (1) is minimized by the Nadaraya-Watson kernel smoother [13]. Minimizing C_{NI} with an MLP leads to have this function as target, which complexity provides an upper bound on the complexity of the trained MLP.

The effective number of parameters p_{eff} of this Nadaraya-Watson smoother can be analytically computed [23]:

$$p_{\text{eff}} = \sum_{i=1}^{\ell} \frac{p_{\boldsymbol{\eta}}(\mathbf{0})}{\sum_{j=1}^{\ell} p_{\boldsymbol{\eta}}(\boldsymbol{x}_i - \boldsymbol{x}_j)} \quad , \tag{18}$$

where $p_{\boldsymbol{\eta}}$ is the noise distribution, acting here as a kernel. The ratio $\sum_{j=1}^{\ell} p_{\boldsymbol{\eta}}(\boldsymbol{x} - \boldsymbol{x}_j)/p_{\boldsymbol{\eta}}(\mathbf{0})$ can be interpreted as a weighted measure of the number of neighbors of $\boldsymbol{x}$, hence p_{eff} is a decreasing function of the average number of neighbors of the training set points.

With Gaussian noise, the isolevels of the weighted neighborhoods define ellipsoids with volume proportional to $V_d\sqrt{\det(\boldsymbol{\Sigma})}$, where V_d is the volume of the unit hyper-sphere. The set of σ_j minimizing the volume of neighborhoods under constraint (4) is the minimizer of:

$$\prod_{j=1}^{d} \sigma_j + \lambda \sum_{j=1}^{d} \frac{1}{\sigma_j^2} \quad ,$$

where λ is the Lagrange multiplier corresponding to constraint (4) for a given isolevel. The solution to this new problem is again $\sigma_j^2 = \sigma^2,\ j = 1 \ldots d$. Thus, elliptical neighborhoods considered by ANI have a larger volume than the spherical neighborhoods of NI.

The increase in neighborhoods volume is likely to increase the average number of neighbors of the training set points, so that, for a given σ^2 value, p_{eff} (18) for ANI should be smaller than in NI. Once again, an upward correction should be applied to take into account the adaptivity of $\boldsymbol{\Sigma}$. However, this rough result supports that the algorithm should be conservative, hence stable, and that complexity can indeed be tuned together with the "effective number of inputs" for non-linear as for linear models.

TABLE I

ANISOTROPIC NOISE INJECTION ALGORITHM FOR VARIABLE RELEVANCE DETERMINATION.

1. Set the value of the hyper-parameter σ^2 and initialize $\sigma_j = \sigma$, $j = 1, \ldots, d$.
2. Optimize the network parameters $\boldsymbol{W}_h$ and θ_h, $h = 1, \ldots, H$ together with $\{\sigma_j^2\}_{j=1}^d$.
   At each epoch, and for each input-output pairs:
   - compute $\boldsymbol{\xi}_i = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}_i$, with $\boldsymbol{\Sigma}_{jj}^{-1/2} = 1/\sigma_j$;
   - feed the net with $\boldsymbol{\xi}_i + \boldsymbol{\eta}$, $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$;
   - compute loss $l(\boldsymbol{f}(\boldsymbol{\xi}_i + \boldsymbol{\eta}), \boldsymbol{y}_i)$;
   - back-propagate the error through the net;
   - update $\boldsymbol{W}_h$ and θ_h, $h = 1, \ldots, H$;
   - back-propagate the error through $\boldsymbol{\Sigma}^{-1/2}$;
   - update $\boldsymbol{\Sigma}^{-1/2}$ with constraint (4).
3. Set $\boldsymbol{W}_1 = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{W}_1$.
4. Estimate generalization error (bootstrap).

TABLE II

RESULTS ON THE ROBOT ARM PROBLEM. RESULTS FOR BAYESIAN LEARNING ARE REPRODUCED FROM NEAL [10].

Model	Average test squared error ($\times 10^{-3}$)				
Bayes, no ARD	5.97				
1-level ARD	5.63				
2-level ARD	5.49				
NI	13.52	12.26	12.24	10.15	13.53
	9.39	16.28	11.05	11.38	17.36
ANI	7.62	7.83	7.96	8.06	8.17
	9.27	9.51	9.53	10.66	10.85

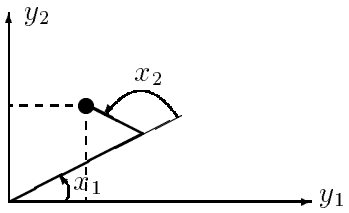

Fig. 1. Robot arm problem.

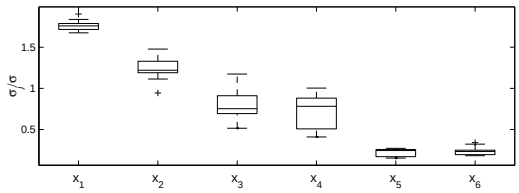

Fig. 2. Relevance indices estimated by ANI for the robot arm problem.

TABLE III

RESULTS ON THE NOISY LED DISPLAY PROBLEM. RESULTS FOR BAYESIAN LEARNING ARE REPRODUCED FROM NEAL [10].

Model	Average test error rate (%)				
Bayes, no ARD	37.8		37.6	33.1	
Bayes, ARD	30.4		31.7	33.0	
NI	33.0	31.9	33.1	36.3	30.5
	30.6	32.0	32.4	32.7	34.0
ANI	25.8	26.0	26.2	26.3	26.4
	26.4	26.8	27.1	27.8	28.0

TABLE IV

AVERAGE TEST ERROR RATES IN GENERALIZATION FOR THE SIX EXAMPLES. CART:LN DENOTES CART WITH LINEAR COMBINATIONS, KNN IS THE K-NEAREST NEIGHBORS ALGORITHM, KNN:FM IS THE BEST OF FIVE VERSIONS OF FLEXIBLE METRIC NEAREST NEIGHBORS ALGORITHMS [19].

Ex.	CART	CART:ln	KNN	KNN:fm	ANI
1	15.1	14.8	13.2	6.3	**2.2**±0.6
2	14.6	11.7	10.3	3.4	**3.0**±0.6
3	21.9	21.8	35.9	**21.7**	25.8±8.5
4	32.2	32.3	34.0	**26.2**	26.9±4.0
5	32.4	7.6	17.4	6.2	**3.2**±1.8
6	29.1	21.1	17.1	18.7	**16.0**±0.7

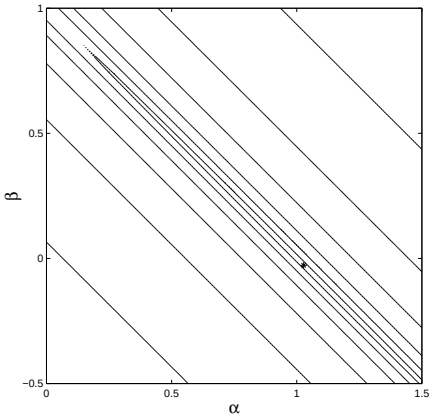

Fig. 3. Contour plot of $C(\alpha, \beta)$ (10). The star represents the optimal value over the 200×200 regular grid in $[0, 1.5] \times [-0.5, 1]$.

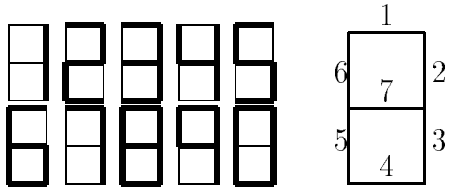

Fig. 4. Digit patterns for the noisy LED display and feature numbers used for coding.

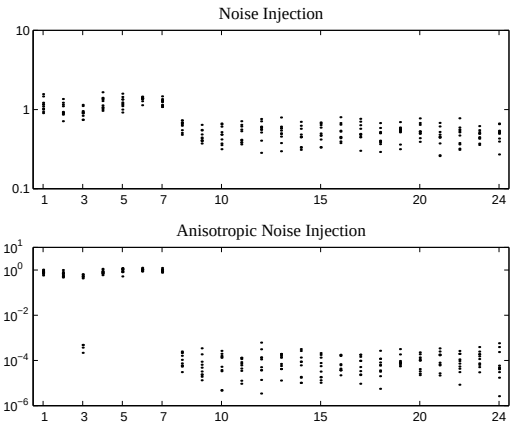

Fig. 5. Square roots of the average squared magnitude of fan-out weights for the twenty-four input units for the solutions obtained from the ten training sets on the noisy LED display problem (log scale).

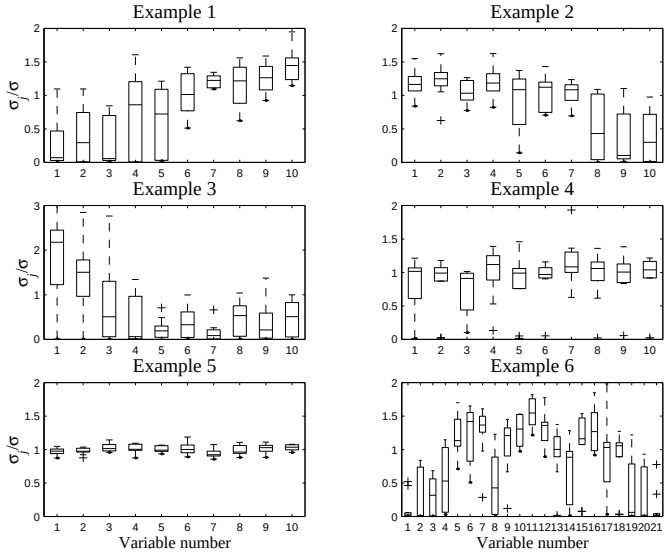

Fig. 6. Relevance indices for the six examples.

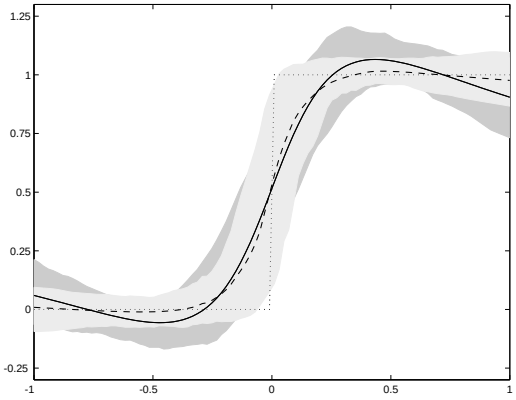

Fig. 7. Average network responses for NI (dashed line) and weight decay (solid line). The dotted line is the target function and the light/dark grey regions represents the centered 80% range of network responses for NI/weight decay.