Relevance Vector Machine and Sparse Vector Regression : a study on the sparsity versus performance tradeoff

Hadrien Hendrikx
School of Computer and Communication Sciences (IC)
École Polytechnique
Fédérale de Lausanne
Lausanne, Switzerland
hadrien.hendrix@epfl.ch

Gregoire Gallois-Montbrun
School of Engineering (STI)
École Polytechnique
Fédérale de Lausanne
Lausanne, Switzerland
gregoire.gallois-montbrun@epfl.ch

Louis Faury
School of Engineering (STI)
École Polytechnique
Fédérale de Lausanne
Lausanne, Switzerland
louis.faury@epfl.ch

***Abstract*—Over the last decade, sparse kernel machines have been at the center of a general interest from both the research community and the industrial one. Once trained, their inherent sparsity allows for fast computations, even on large datasets.**
This report, written in the context of a machine learning coding project, compares two sparse regression techniques.
After introducing some theoretical aspects on Support Vector Regression (SVR) and the Relevance Vector Regression (RVR), we introduce two datasets (an artificial one well as a real one) that we will use as baselines for experimental comparisons between the two methods.
We will then focus the comparison on a few key features. Namely, we will analyze the tradeoff found between generalization and sparsity in the two methods. Other issues such as robustness (behavior far from data, dataset scavenging and data chunking) and complexity will also be tackled.

I. Introduction

We hereinafter focus on regression tasks : from a dataset $\{X, \mathbf{t}\} \in \mathbb{R}^{N,d} \times \mathbb{R}^N$, we wish to learn a function f such that $f(X) \simeq \mathbf{t}$.
The simplest method to perform such a task is to consider that $f(\cdot)$ is a linear function of a non-linear mapping of the input x, corrupted by a normally distributed noise of precision β

$$t = w^T\phi(x) + \eta, \quad \eta \sim \mathcal{N}\left(\eta \,\middle|\, 0, \beta^{-1}\right) \tag{1}$$

with $\phi : \mathbb{R}^d \to \mathbb{R}^m$ a non-linear transformation. Therefore, $p(t \,|\, w, \beta) \sim \mathcal{N}\left(t \,\middle|\, w^T\phi(x), \beta^{-1}\right)$ and the *maximum-likelihood* solution is given by the minimizer of the opposite of the log-likelihood :

$$w_{MLE} = \operatorname*{argmin}_w \left\{ \frac{1}{2}\sum_{n=1}^{N} \left\{ w^T\phi(x_n) - t_n \right\}^2 \right\} \tag{2}$$

To avoid overfitting, a regularizer term is often added to the cost function :

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} \left\{ w^T\phi(x_n) - t_n \right\}^2 + \frac{\lambda}{2}\|w\|^2 \tag{3}$$

even if some more advanced methods (Bayesian learning for instance) can also be used to avoid prevent data overfit.

II. Theoretical background

A. Support Vector machine for Regression

The Support Vector machine for Regression (SVR) extends the SVM method for regression tasks.
To obtain a sparse solution, the likelihood term in (3) is replaced by an *ε-insensitive error function* (see [7]) denoted $E_\varepsilon(\cdot)$ with :

$$E_\varepsilon(y(x) - t) = \begin{cases} 0, & \text{if } |y(x) - t| < \varepsilon \\ |y(x) - t| & \text{otherwise} \end{cases} \tag{4}$$

Therefore, the quantity to be minimized can be expressed as :

$$J(w) = \frac{C}{n}\sum_{n=1}^{N} E_\varepsilon(w^T\phi(x_n) - t_n) + \frac{1}{2}\|w\|^2 \tag{5}$$

where C is a regularization parameter.
As for the SVM, one can introduce *slack variables* in order to transform this optimization program into a Quadratic Programming (QP) problem (quadratic objective, linear constraints) :

$$\begin{aligned} \min_{w,\, b} \quad & \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\left(\xi_i + \hat{\xi}_i\right) \\ \text{s.t.} \quad & \xi, \hat{\xi} \geq 0 \qquad \text{(row wise)}\,, \\ & w^T\phi(x_i) + b + \varepsilon + \xi_i \geq t_i, \quad i = 1, \ldots, n, \\ & w^T\phi(x_i) + b - \varepsilon - \hat{\xi}_i \leq t_i, \quad i = 1, \ldots, n \end{aligned} \tag{6}$$

Once the problem solved, predictions are made using :

$$y(x) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(x, x_n) + b \tag{7}$$

where we introduced the kernel function $k(x, x') = \phi(x)^T\phi(x')$. The coefficients $\{a_n\}$ and $\{\hat{a}_n\}$ actually are Lagrange multipliers for the QP problem (6), and provide a *sparse* solution - only a few data points are used for regression. Those are called *support vectors*, and are so that $a_n \neq 0$ or

$\hat{a}_n \neq 0$ - in other words, those that lie on the boundary or outside of the ε-tube defined by the loss function in equation (4).
As for the SVM, one can adopt a ν-SVR formulation (see [4]) to have a *lower-bound control* on the number of retained support vectors.

B. Relevance Vector machine for Regression

The SVR therefore provides a useful tool for obtaining sparse regression machine. However, it suffers from a *number of limitations*, such as an output representation as decision rather than posterior probability, the need to estimate hyper-parameters (kernel width, penalization parameter) via *held-out methods* (like cross-validation), or the need for the kernel to be a Mercer kernel type (positive definite).
The Relevance Vector Machine for regression is a *Bayesian sparse kernel technique* that shares many of the SVR's characteristics while avoiding its limitations. It instantiates a model intended to mirror the structure of the SVR :

$$y(x) = \sum_{n=1}^{N+1} w_n k(x, x_n) \tag{8}$$

where the bias b is included in the predictor w and $k(\cdot,\cdot)$ is an arbitrary kernel (not necessarily positive definite). Assuming i.i.d data sample with Gaussian noise of precision β, the likelihood writes :

$$p(\mathbf{t} \,|\, X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n \,\middle|\, y(x_n), \beta\right) \tag{9}$$

The predictor w is given a centered Gaussian prior distribution :

$$p(w \,|\, \alpha) = \prod_{i=1}^{N+1} \mathcal{N}\left(w_i \,\middle|\, 0, \alpha_i^{-1}\right) \tag{10}$$

introducing a separate precision parameter α_i for each weight parameter w_i.
This leads to a Gaussian posterior distribution over w :

$$\begin{aligned} p(w \,|\, \mathbf{t}, X, \alpha, \beta) &= p(\mathbf{t} \,|\, w, X, \beta) p(w \,|\, \alpha) \\ &= \mathcal{N}\left(w \,\middle|\, m, \Sigma\right) \end{aligned} \tag{11}$$

where

$$\begin{aligned} m &= \beta \Sigma \boldsymbol{\phi}^T t \\ \Sigma &= \left(A + \beta \boldsymbol{\phi}^T \boldsymbol{\phi}\right)^{-1}, \qquad A = \text{diag}(\alpha_i) \end{aligned} \tag{12}$$

In a full Bayesian approach, α and β are both given prior distributions. However, this leads to intractable computations when computing predictions. The use of *empirical Bayes* solves this problem, by approximating α and β by their maximum-a-posteriori value (also known as the *evidence approximation* or type-2 maximum likelihood).
As a result of approximation, a proportion of parameters α_i are driven to infinite values, constraining the corresponding weights w_i to have 0 mean and infinite precision, and hence are set to 0. The resulting predictions are therefore sparse in datapoints, and the inputs $\{x_n\}$ corresponding to non-zero weights are called *relevance vectors*. Once the optimal values α^* and β^* found, the predictive distribution over y can therefore be computed using α^* and β^*.
The sparsity analysis of the RVR leads to a practical algorithm for optimizing the hyper-parameters that has significant speed advantages, and is referred to as *automatic relevance determination*. The full algorithm and its justification can be found in [1] and [6].
Furthermore, the RVR provides a posterior distribution over the predictor, leading to the following *predictive distribution* (unlike the SVR which only provide a posterior decision) :

$$\begin{aligned} p(t \,|\, x, X, \mathbf{t}, \alpha, \beta) &= \int_w p(t \,|\, w, x, \beta) p(w \,|\, X, \mathbf{t}, \alpha) dw \\ &= \mathcal{N}\left(t \,\middle|\, m^T \phi(x), \beta^{-1} + \phi(x)^T \Sigma \phi(x)\right) \end{aligned} \tag{13}$$

with m and Σ the first moments of the posterior (see (12))

C. Theoretical method comparaison

1) Complexity: When comparing the SVR's and the RVR's complexity, one must distinguish complexity at training time and at testing time.
Training the SVR sums up in solving a large quadratic-programing (QP) problem. A popular approach to do that implies breaking up the initial QP into smaller problems, solvable analytically, and is called *Sequential Minimization Optimization* (SMO). It requires a linear (with respect to the datapoints) amount of memory, and scales between linear and quadratic complexity in the training set size (see [3]). At test time, the complexity is linear in the number of support vectors.
Training a RVM involves optimizing a non-convex function. For a model with M basis functions, the RVM requires the inversion of a $M \times M$ matrix, which requires from $O(M^{2.7})$ to $O(M^3)$, which is as we just saw larger than the SVR's cost. However, parameters are determined automatically and in one run when training a RVR, while the hyper-parameters typically need several runs (f-fold cross-validation) to be estimated. At testing time, the RVR's complexity grows linearly with the number of relevance vectors.

2) Performance: We just named one of the major pros of using RVR - there is no need for using held-out methods to estimate hyper-parameters, as they are automatically determined through automatic relevance detection (expect for parametric basis functions). Also, it has been shown [1], [3] that RVR leads to *sparser solutions, without loss of generalization abilities* (on the contrary, the RVR usually performs better than the SVR). We will try to observe this in the experiments derived hereinafter.

III. Experimental results

A. Datasets presentation

1) Artificial dataset: The goal of using an artificial dataset is to be able to produce visual outputs to evaluate the performance of the two methods on a simple regression problem, hence the need to work in a one dimensional dataset.
The dataset was generated from the $sinc(\cdot)$ function, on the

interval $[-5, 5]$. Points were randomly sampled from this interval, applied the $sinc$ function and added a Gaussian distributed centered noise of variance $\sigma^2 = 0.01$.
The following table sums up the dataset characteristics.

Dimension	**Points**	**Support**	**Noise variance**	**Outlier**
1	100	[-5,5]	0.01	No

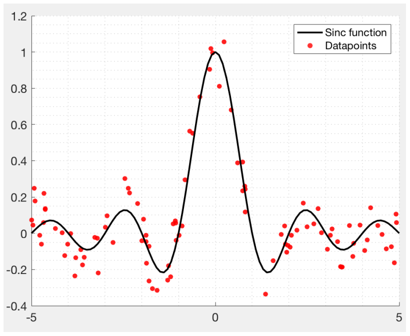

Fig. 1: The 1D artificial dataset

2) Real dataset: This dataset, known as the Airfoils Self-Noise dataset provides informations about design of various airfoils and physical environment (free stream velocity, angle of attack, chord length, frequency, suction side thickness) in order to predict the resulting sound pressure level in dB (see [5] for full details). The following table sums up the dataset characteristics. All experiments were performed after standardization of the dataset.

Dimension	**Points**
5	1503

B. Implementation

We hereinafter describe different computations and results obtained on the artificial data with both the SVR and the RVR. Discussion on the two methods advantages and drawbacks will be held in the next section.

1) Support Vector Regression: We implemented both the ε-SVR and the ν-SVR on the artificial dataset. We used the `libsvm` library (see [2]) with a radial-basis function kernel :

$$\forall x, x', \quad k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{\sigma^2}\right) \tag{14}$$

We therefore tuned three hyper-parameters : σ (kernel width), C (penalization factor) and ν (resp. ε) for the ν-SVR (resp. ε-SVR). Figures (3) and (4) display the regressive function obtained for three given parameters for both methods on the artificial dataset. For the ν-SVR, the corresponding ε-insensitive tube was computed thanks to the values of the final Lagrange multipliers (a Lagrange multiplier strictly inferior to C indicates a point laying on the ε-tube).

2) Relevance Vector Regression: A similar work was conducted with the RVR, by using the `SparseBayes` library written by Mike Tipping. The same kernel (RBF) was used. Therefore, the only hyper-parameter to be fixed is the kernel's width σ (since automatic relevance detection automatically finds the best β and α). For regression, we plot the mean

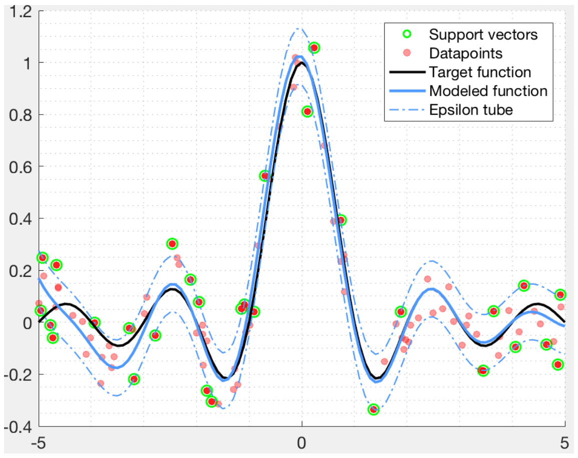

Fig. 2: ν-SVR with : $C = 100$, $\nu = 0.2$, $\sigma = 1.5$

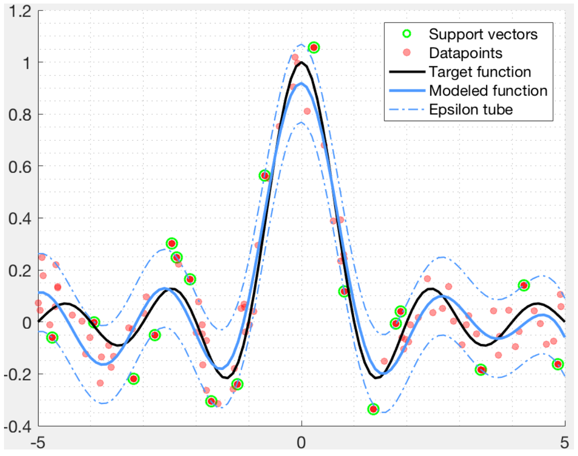

Fig. 3: ε-SVR with : $C = 50$, $\varepsilon = 0.15$, $\sigma = 2$

of the *predictive distribution* as well as it's standard deviation (see (13)).

C. Cross-validation

1) A sparse regression evaluative metric: To infer the best hyper-parameters for all three algorithms (ϵ-SVR, ν-SVR and RVR), grid-search cross-validation is used. The metric we used to evaluate performance of the algorithms is derived from the BIC (Bayesian Information Criterion, see [1]). Among the properties of this metric is the ability to evaluate a compromise between complexity and likelihood of different models. We want to reproduce this characteristic to evaluate the performance versus sparsity tradeoff found by different models. The following lines intent to motivate the metric we chose to use.
The general formula of the BIC metric is:

$$BIC = -2ln(L) + kln(N) \tag{15}$$

where L corresponds to the likelihood of the model, k is the number of parameters of the model and N is the number of data-points. In the particular case we wish to tackle (*i.e*

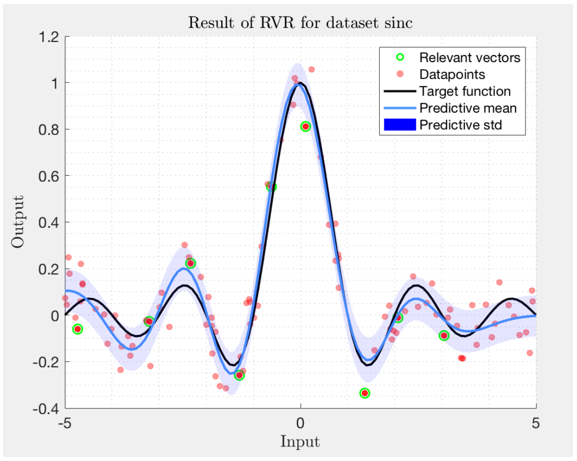

Fig. 4: RVR with : $\sigma = 1$

measuring a trade-off between regression performance and sparsity), the BIC appears to be a reasonable metric, although some adjustments are needed to adapt it to a regression task. Assuming independent, normally distributed noise, the likelihood can naturally be related to the Mean Square Error (MSE) :

$$ln(L) = -\frac{N}{2\sigma^2}MSE \tag{16}$$

where σ^2 corresponds to the variance of the dataset's noise.
For both algorithms, the number of parameters k is directly proportional to the number of relevant vectors (respectively support vectors). Therefore a reasonable metric to measure the goodness of a sparse regression model would be

$$BICSR = N\frac{MSE}{\sigma^2} + N_{SV}ln(N) \tag{17}$$

where N_{SV} is the number of relevance (or support) vectors, and $BICSR$ holds for Bayesian Information Criterion for Sparse Regression. Minimizing this metric indeed implies minimizing the MSE while linearly penalizing the number of support vectors. Some other kind of penalization (quadratic, exponential) could also be used. The discussion of the choice of this penalization, as well as the validation of the well behavior of this metric is held in (IV).

We wish to emphasize the fact that this metric is not the commonly used BIC metric, but rather an adaptation of it to sparse regression (and therefore a measure of trade-off between performance (MSE) and sparsity).

2) *Grid-search cross-validation:* In the following, we performed 50-fold cross-validation with a 0.75 training-test ratio, using the $BICSR$ previously defined.

There are three hyperparameters to estimate for ϵ-SVR. ϵ represents the width of the insensitive tube, and its optimal value should be close to the actual value for standard deviation of the output noise. The explored range [0.001, 10] seemed therefore adequate. C penalizes for distances between points lying outside the ϵ-tube and the border of the tube, it is hard to estimate beforehand the amount of order of the optimal value

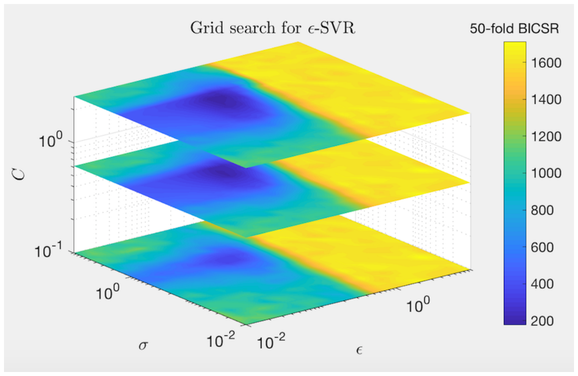

Fig. 5: Grid search for ϵ-SVR over ϵ, C and σ for sinc dataset
Best hyperparameters: $\epsilon = 0.17$, $C = 2.40$, $\sigma = 1.17$

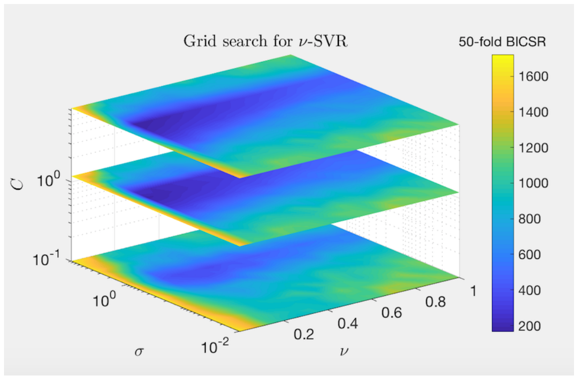

Fig. 6: Grid search for ν-SVR over ν, C and σ for sinc dataset
Best hyperparameters: $\nu = 0.057$, $C = 1.52$, $\sigma = 1.13$

so that grid search was performed on the interval [0.1, 1000]. Finally, the kernel width σ represents the region of influence of the different support vectors. Given the spacing of inputs for the dataset, a good value should lie near 1 so that explored range was set to [0.001, 10].
For ν-SVR, ν necessarily lies in the interval]0,1[, the explored range, and represents a lower bound on the fraction of support vectors. Other considerations for kernel width σ and penalization factor C still hold.

Figures (5) and (6) display grid search results for both ϵ-SVR and ν-SVR in the form of heat-maps. If we fix C, we can distinctively see that the objective function forms a well. Changing the value of C only changes the size of the well and its value but not the global behaviour, which suggests that this parameter has little influence on the result as long as values are not too far from the optimal. This leads us to believe that there is a global minimum for our objective function, that this minimum is captured by our grid and therefore that the result of our grid search is reliable. The form of the regions is slightly different between ϵ-SVR and ν-SVR but the general behaviour are the same.

For ϵ-SVR, the plateau corresponding to highest values of ϵ corresponds to models where all points lie inside the ϵ-tube. Finally, as expected, optimal models for both methods are very

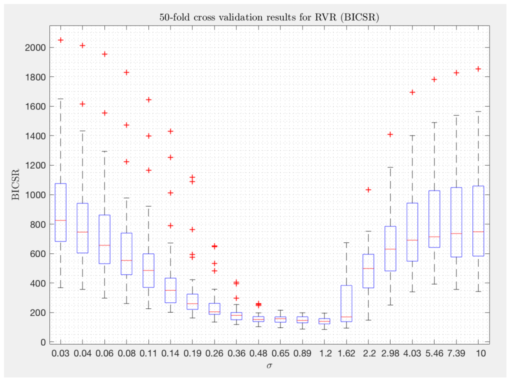

Fig. 7: Grid search for RVR over σ for sinc dataset. Best hyperparameter: $\sigma = 1.28$

close, since they share most of their support vectors, therefore showing the equivalence between both approaches. Both lead to a total of 14 support vectors, corresponding to a Mean square error of 0.013.

Figure (7) displays grid-search results for the RVR over kernel width σ, the only hyperparameter for this method. Found optimal value 1.28 again confirms expectations that the order of magnitude is well chosen since it is close to the value chosen for SVR methods. Corresponding number of relevant vectors is eight, leading to a MSE of 0.01. As a result, for this particular metric which demonstrates good properties concerning the sparsity/accuracy trade-off, as will be demonstrated in section (IV), the RVR optimal model leads to a sparser and more accurate solution than optimal SVR models.

D. Real dataset

For this dataset, no prior knowledge on the variance of the noise affecting the outputs was available. As a consequence, it had to be estimated before applying $BICSR$ metric to the dataset (see section (IV)). As we have seen in the previous study, Relevance Vectors Regression always provides sparse solutions while estimating output noise via parameter β. To find a relevant value for kernel width, a first grid search was therefore performed for RVR with MSE. Corresponding estimation for β (equals to 2.4) was then used to evaluate $BICSR$ for all models. We then performed grid search using 10-fold cross validation to estimate best values for algorithms hyperparameters. Due to higher dimensionality and to an higher number of points, training test ratio was decreased to 0.5.

Figures (8) and (9) expose obtained results for both nu-SVR and ϵ-SVR. General behavior of the algorithms to the different hyperparameters variations seemed consistent with observations made on the artificial sinc dataset. Again, distinct wells could be observed when displaying the objective function evolution leading to the conclusion that found optimums are global. For ϵ-SVR, the optimal hyperparameters values lead to a Mean Square Error of 0.16 for 295 support vectors while ν-SVR yields 249 support vectors for a MSE of 0.18.

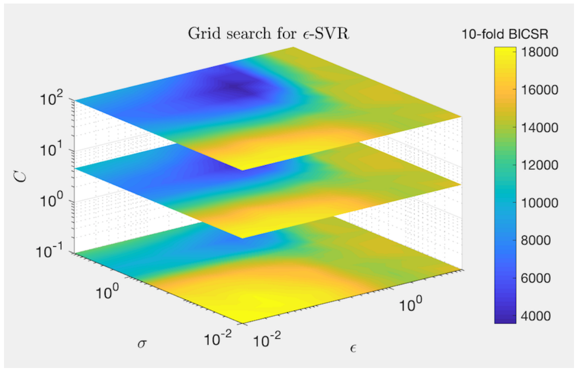

Fig. 8: Grid search for ϵ-SVR over ϵ, C and σ for airfoils dataset Best hyperparameters: $\epsilon = 0.46$, $C = 146.8$, $\sigma = 2.15$

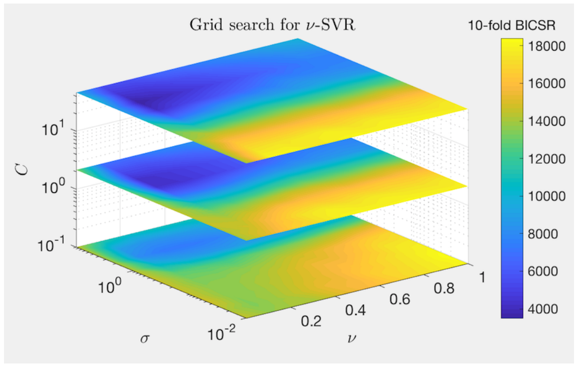

Fig. 9: Grid search for ν-SVR over ν, C and σ for airfoils dataset Best hyperparameters: $\nu = 0.11$, $C = 46.4$, $\sigma = 2.15$

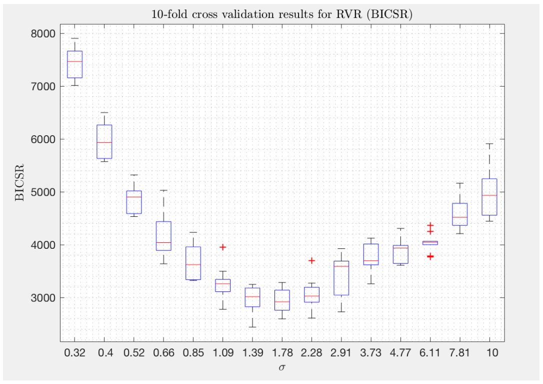

Fig. 10: Grid search for RVR over σ for airfoils dataset. Best hyperparameter: $\sigma = 1.78$

Results of grid search over σ, the kernel width for Relevance Vectors Regression are displayed on figure (10). Optimal kernel width therefore corresponds to $\sigma = 1.78$ leading to an MSE of 0.17. Number of retrieved relevance vectors is 66. Again, for this real dataset, the RVR optimum leads to relatively equivalent generalization performance but with a lot more sparsity.

IV. Discussion

A. The BICSR metric

The use of the hand-designed BICSR metric deserves some further justification. As a reminder, the BICSR writes :

$$BICSR = N\frac{MSE}{\sigma^2} + kln(N) \tag{18}$$

with N the number of datapoints, MSE the mean-squared error on the dataset, σ^2 the dataset's noise variance and k the number of support vectors.

1) Model likelihood estimation: When applied to an artificial dataset, the noise variance σ^2 is known. However, this parameter must be estimated for any arbitrary dataset. Hence, in order to find, for a given model, its related optimal hyper-parameter (with respect to the BICSR), we first need, equivalently, to estimate the noise precision coefficient $\beta = \frac{1}{\sigma^2}$. The RVR already provides a rather robust estimate of β, without bootstrapping from any initial value (automatic relevance determination), and without need for much parameters tuning. We therefore suggest the following approach for determining β :

1) Run a cross-validation for the RVR with a the MSE metric on the dataset
2) Identify the optimal kernel-width (or equivalently any given kernel hyperparameter)
3) Set $\sigma = \frac{1}{\sqrt{\beta}}$ with β the precision given by the RVR with the previously defined kernel width

To justify the use of this method, we applied it on our artificial dataset, and measured an empirical noise variance $\sigma^2 = 0.008$ instead of 0.01.

2) BICSR validation: To further justify this approach, we wanted to evaluate the tradeoff found by the BICSR. To do this, we cross-validate (10 folds, 0.75 training-test ratio) several ν-SVR arbitrary models (hence with arbitrary hyper-parameters - kernel width, ν and C) and plot them on a MSE versus fraction of support vectors figure. We repeated this experiment with the models generated by the optimal BICSR and MSE metrics. As shown in (11), the BICSR optimal model can be found at the elbow of the MSE vs sparsity curve.

As previously introduced, the BICSR metric measures the tradeoff between MSE and sparsity. It linearly penalizes the number of support vectors, proportionally to lnN. We wondered if some other (utlimately better, depending on the end goal) tradeoff could be found by changing the nature of this penalization. For instance, we could think of a non-linear penalization of the numbers of support vectors (a quadratic penalization would lead to sparser models, and a logarithmic one to more accurate ones). Figure (12) displays the MSE

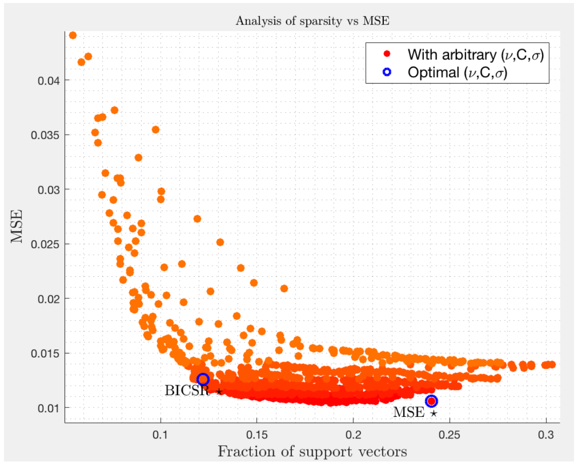

Fig. 11: BICSR and MSE optimal models evaluation

versus sparsity models found when varying the nature of the penalization. Hence, it seems that our initial metric (designated as BISCR) naturally finds a rather interesting tradeoff for generalization against sparsity. We will therefore use it as our baseline metric for the different model comparaisons.

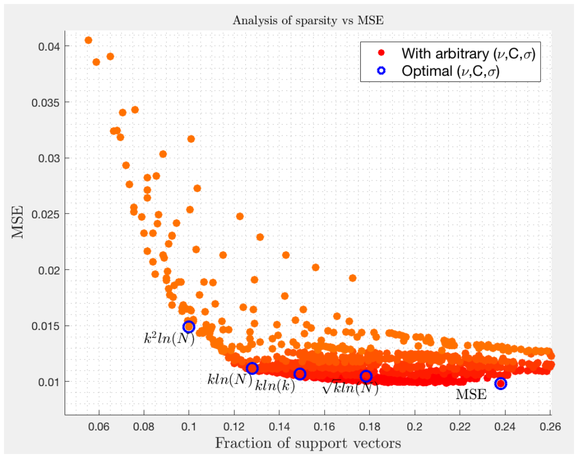

Fig. 12: Evaluation for different penalizing terms

3) Model comparaison: Now that we have more insights into how our metrics behaves, it is possible to compare the different models that maximize it.

Figure (13) allows us to see how the different methods react to the trade-off sparsity versus performance (which is measured by MSE minimization).

First of all, we see that except from the ε-SVR maximizing the BICSR, all MSE values are comparable (around 15% difference). However, there is a big gap in terms of sparsity since SVR with the MSE metrics have about twice as much support vectors as SVR that minimizes the BICSR metric.

The key insights from this figure are that SVR methods with MSE objective have many support vectors, and that the BICSR metric leads to sparser solutions with minimal increase of the MSE. RVR achieves the best sparsity by far with minimal

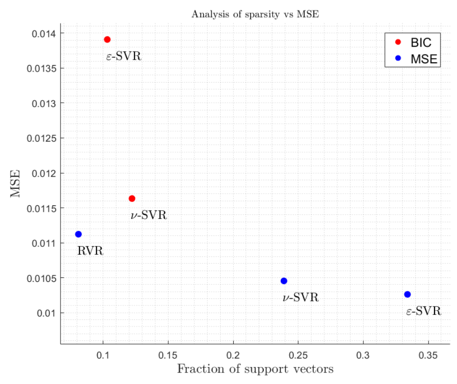

Fig. 13: Evaluation of the tradeoff between raw performance (MSE) and for each method and each metric for the sinc dataset.

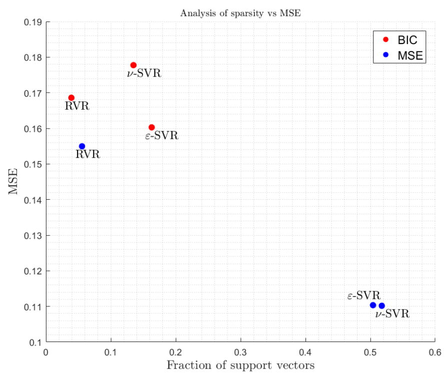

Fig. 14: Evaluation of the tradeoff between raw performance (MSE) and for each method and each metric for the airfoils dataset.

MSE increase and there is almost no difference between the models selected by the two different metrics.

The sinc dataset was a bit too easy for our methods, which is why MSE values were close. Figure (14) shows the same curve as before but for the airfoils dataset. We can observe now that the difference in terms of MSE is significant (up to 70%). The difference in sparsity is also huge since SVR solutions minimizing the MSE have 10 times more support vectors than RVR solutions (50% of the dataset versus 5%). We see that ν-SVR and ε-SVR perform relatively similarly (although there is a little difference for the BICSR metric), and that they are very sensitive to the change of metric. On the other hand, RVR solutions are close to solutions for SVR using the BICSR metric. Again, RVR is not very sensitive to the metric used although there is now a difference between the two.

To summarize, RVR should be prefered when sparsity is a key feature because it always leads to sparser solutions than SVR (with better MSE for comparable levels of sparsity) and is less sensitive to the choice of the metric. On the other hand, this insensivity may be a problem since it only leads to very sparse solutions. To find the best MSE while staying under a relatively high given threshold of support vectors (0.3 for example) then it is possible to tweak a metric (BICSR with sublinear penalization on the number of support vectors for example) and then to use SVR, whereas RVR would almost always give the same result.

Not that here we use the same set for cross-validation and testing. This may lead the hyper-parameters to overfit the cross-validation set, therefore leading to lower scores than one would expect at generalization. However, this is not critical for our study since we are more interested in the impact of the metrics and not the true generalization of our methods.

B. *Behavior far from data*

If we focussed until now on the tradeoff between sparsity and generalization performances, their are other aspects worthy of discussion. Namely, the two considered methods (SVR and RVR) could be compared on the basis of *generalization far from the data.*

On this particular question, both methods show the same drawbacks. Indeed, the predicted function far from data is limited to the learned bias (at least for centered kernels, like the Gaussian RBF). One could argue that this drawback is limited for the RVR, as the variance of the predictive distribution should grow as we get far from the data, indicating that predictions are not to be trusted with. However, (13) proves that this assumption does not hold, as for the RVR the predictive distribution is higher close from the data, and constant far from it.

Therefore, both methods are not to be trusted with far from the data (they only capture the regressive function's mean and not the general trend). Other Bayesian methods, like Gaussian Process Regression would probably do a better job at this particular task - but use the entire dataset when making predictions.

V. Conclusion

$$ln(L(\mathbf{t}, X, w, \beta)) = -\frac{1}{2}\beta^2 \sum_{i=1} N(t_n - y(x_n))^2 \qquad (19)$$

where we recall that estimated parameter β is supposed to represent the precision (inverse of the variance) of the noise affecting the output. Therefore, a

References

[1] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[3] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[4] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[5] D.S. Pope T.F. Brooks and A.M. Marcolini. Airfoil self-noise and prediction. *Technical report, NASA RP-1218*, 1989.

[6] Michael Tipping. Relevance vector machine, October 14 2003. US Patent 6,633,857.

[7] Vladimir N Vapnik. The nature of statistical learning theory. 1995.