Relevance Vector Machine for regression and comparaison with the SVR

Hadrien Hendrikx
School of Engineering (STI)
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*—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 as well as a real one) that will serve us as baselines for experimental comparisons between the two methods.
We will then focus the comparison on a few key features. Namely, we will study the tradeoff found between generalization and sparsity in the two methods. Issues such as robustness (behavior far from data, dataset scavenging and data chunking), computational complexity and training / testing time 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

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

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

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

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 [5]) denoted $E_\varepsilon(\cdot)$ with :

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

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) :

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

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 $k(x, x') = \phi(x)^T\phi(x')$. 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

[3]) 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 :

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

The predictor w is given a centered Gaussian prior distribution :

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

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

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

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 [4].
Furthermore, the RVR provides a posterior distribution over the predictor, leading to

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 [2]). 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], [2] 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 observations in the experiments derived hereinafter.

III. 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 normally distributed noise. For reasons we'll tackle later, no outliers were added in the dataset.
The following table sums up the dataset characteristics.

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

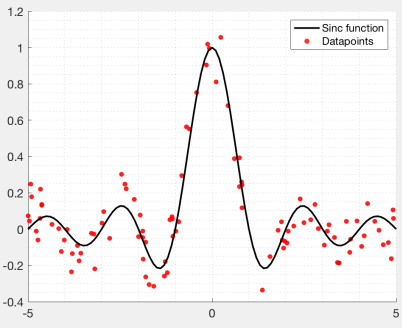

Fig. 1. The 1D artificial dataset

B. Results

IV. Discussion

V. Conclusion

References

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

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

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

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

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