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*—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.**

I. Introduction

II. Theoretical background

A. Support Vector machine for Regression

The Support Vector machine for Regression (SVR) extends the SVM method for regression tasks. Let us first consider the simple linear regression case for a dataset $\{X, \mathbf{t}\} = \{(x_1, \dots, x_N)^T, (t_1, \dots, t_n)^T\}$, where we minimize a regularized error function given by :

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

To obtain sparse solution, the quadratic term is replaced by a *ε-insensitive error function* (see [3]) denoted $E_\varepsilon(\cdot)$ with :

$$E_\varepsilon(y(x) - t) = \begin{cases} 0, & \text{if } |y(x) - t| < \varepsilon \\ |y(x) - t| < \varepsilon & \text{otherwise} \end{cases} \tag{2}$$

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

$$C\sum_{n=1}^{N} E_\varepsilon(w^T\phi(x_n) - t) + \frac{1}{2}\|w\|^2 \tag{3}$$

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 problem (quadratic objective, linear constraints). Once the problem solved, predictions are made using :

$$y(x) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(x, x_n) + b \tag{4}$$

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, and provide a *sparse* solutions in the data-points. The *support vectors* (data point used for predictions) are those for which $a_n \neq 0$ 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 (2).
As for the SVM, one can adopt a ν-SVR formulation to have an 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 :

$$y(x) = \sum_{n=1}^{N+1} w_n k(x, x_n) \tag{5}$$

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 :

$$p(\mathbf{t} \,|\, X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n \;|\, y(x_n), \beta\,\right) \tag{6}$$

The predictor w is given a centered Gaussian prior distribution :

$$p(w \mid \alpha) = \prod_{i=1}^{N+1} \mathcal{N}\left(w \mid 0, \alpha_i^{-1}\right) \tag{7}$$

introducing a separate precision parameter α_i for each weight parameter w_i.

This leads to a Gaussian posterior distribution over w :

$$\begin{aligned} p(w \mid \mathbf{t}, X, \alpha, \beta) &= p(\mathbf{t} \mid w, X, \beta) p(w \mid \alpha) \\ &= \mathcal{N}\left(w \mid m, \Sigma\right) \end{aligned} \tag{8}$$

where

$$\begin{aligned} m &= \beta \Sigma \boldsymbol{\phi}^T t \\ \Sigma &= \left(A + \beta \boldsymbol{\phi}^T \boldsymbol{\phi}\right)^{-1}, \qquad A = \mathrm{diag}(\alpha_i) \end{aligned} \tag{9}$$

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*).

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].

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 in 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.

D. 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 ([2]), [1]) that RVR leads to *sparser solutions, without loss of generalization abilities* (on the contrary, the RVR usually perform better than the SVR). We will try to observe this observations in the experiments derived hereinafter.

III. Results

A. Datasets presentation

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