Testing Support Vector Interpolation for GSP-Phot forward-model interpolation

prepared by: René Andrae
reference: GAIA-C8-TN-MPIA-RAN-029
issue: 1
revision: 1
date: 2015-10-30
status: Issued

Abstract

Given the usually excellent performance of support vector machines in classification and regression, we devise an algorithm for support vector interpolation. We test whether such an SVI is useful for replacing the currently implemented GSP-Phot forward-model interpolation. Our focus is on testing whether support vector interpolation can reliably predict BP/RP spectra with extinction, using smaller model grids which consume less memory. Unfortunately, we find the results from support vector interpolation are not good enough to compete with linear interpolation.

Document History

Issue	Revision	Date	Author	Comment
1	1	2015-10-30	RAN	First issue.
D	1	2015-10-30	RAN	Including comments from Coryn.
D	0	2015-10-08	RAN	First draft.

1 Introduction

Support vector machines (SVM, e.g., Deng et al., 2012) are a class of very popular off-the-shelf learning algorithm applied in classification, regression and density estimation. SVMs typically have excellent performance and require little or no fine-tuning to the specific problem at hand. We therefore want to devise an algorithm for support vector interpolation (SVI).

Our primary objective is to find out whether such an SVI can provide a more reliable interpolation over extinction, A_0, than linear interpolation. In particular, we want to test whether we can use SVI to replace the GSP-Phot forward model, which is currently implemented as linear interpolation (RAN-025). This linear interpolation was found to require extremely dense model grids in order to avoid outliers caused by inaccurate interpolation results (RAN-027). The reason is that extinction has a highly nonlinear impact on the pixel flux. Therefore, an SVI with a properly tuned kernel function could provide better results. This might help to avoid forward model grids which are overly dense in A_0 and thus require too much memory. As GSP-Phot was found to exceed the DPCC memory limitations during OR5 stage 3, SVI could potentially be of help.

As a first guess, a support vector regression (e.g., RAN-024) could be employed to predict the flux in some spectral pixel for given stellar parameters. However, using such a regression would allow the model function to not precisely match the given training spectra, which we consider undesirable in an interpolation context. Instead, we need to impose the perfect matching of the given grid spectra, which is the key difference between SVI and support vector *regression*.[1] Nevertheless, any SVI algorithm will naturally exhibit close similarities to support vector regression.

2 Mathematics

In this section, we are going to derive the equations of SVI. We did not find any mentioning of SVI in the scientific literature, so in that sense, these elaborations are “novel”. However, the

[1] In that sense, we could employ SVR as a “smoothing” algorithm, like a thin-plate-smoothing spline. But for the GSP-Phot forward model, we seek an interpolation method.

maths of SVI are highly similar to those of least-squares SVR (Deng et al., 2012) to the degree that SVI can be considered as a special case of LSSVR.

Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ be a given set of N observed values at locations $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, …, $\boldsymbol{x}_N$. The goal of support vector interpolation is to find a linear hyperplane,

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b \,, \tag{1}$$

which fits all of the N points *perfectly*. Here we assume that it is actually possible for $f(\boldsymbol{x})$ to fit all the given training data perfectly. If that is possible or not depends entirely on the dimensionality D of the input space in which the $\boldsymbol{x}_n$ live. The linear hyperplane given by $f(x)$ can fit all given data perfectly, if and only if $D > N$. However, we will adopt some kernel function $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$, which maps the input space into some higher-dimensional feature space. In fact, the kernel function we will choose creates a feature space of *infinite* dimension, such that $D > N$ is always satisfied. Therefore, the linear hyperplane of Eq. (1) does not apply to the input space but rather to the feature space.

In close resemblance to support vector regression (e.g., Eq. (4) in RAN-024), the objective function reads

$$L = \frac{1}{2}\boldsymbol{w}^2 + \sum_{n=1}^{N} \alpha_n \left(y_n - \boldsymbol{w} \cdot \boldsymbol{x}_n - b\right) \,, \tag{2}$$

where minimisation of the first term as usual maximises the margin width, whereas the second term guarantees the perfect fit to all N observations. Note the absence of any cost parameter C or insensitivity parameter ϵ from Eq. (2), which usually appear in support vector regression. This is simply due to the fact that interpolation requires a perfect fit whereas SVR require C (and sometimes also ϵ) to penalise (and define) residuals.

We proceed as usual and minimise the Lagrangian of Eq. (2) w.r.t. $\boldsymbol{w}$ and b. First, we obtain

$$\boldsymbol{\nabla}_{\boldsymbol{w}} L = \boldsymbol{w} - \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n = 0 \quad \Leftrightarrow \quad \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \,, \tag{3}$$

which is the usual result. Second, we obtain

$$\frac{\partial L}{\partial b} = -\sum_{n=1}^{N} \alpha_n = 0 \quad \Leftrightarrow \quad \sum_{n=1}^{N} \alpha_n = \mathbf{1} \cdot \boldsymbol{\alpha} = 0 \,, \tag{4}$$

which is the same result as for least-squares SVR. If we insert these two findings back into Eq. (2), we get

$$L = \frac{1}{2}\boldsymbol{w}^2 + \sum_{n=1}^{N} \alpha_n y_n - \boldsymbol{w} \cdot \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n - b \sum_{n=1}^{N} \alpha_n \tag{5}$$

$$= \frac{1}{2}\boldsymbol{w} \cdot \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n + \sum_{n=1}^{N} \alpha_n y_n - \boldsymbol{w} \cdot \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \tag{6}$$

$$= -\frac{1}{2}\boldsymbol{w} \cdot \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n + \sum_{n=1}^{N} \alpha_n y_n \tag{7}$$

$$= -\frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n \boldsymbol{x}_m \cdot \boldsymbol{x}_n + \sum_{n=1}^{N} \alpha_n y_n \ . \tag{8}$$

If we introduce matrix notation and re-impose the constraint $\mathbf{1} \cdot \boldsymbol{\alpha} = 0$, we obtain

$$L = -\frac{1}{2}\boldsymbol{\alpha}^T \cdot K \cdot \boldsymbol{\alpha} + \boldsymbol{y} \cdot \boldsymbol{\alpha} - \lambda \mathbf{1} \cdot \boldsymbol{\alpha} \ , \tag{9}$$

where K denotes the kernel matrix whose elements are given by

$$K_{mn} = k(\boldsymbol{x}_m, \boldsymbol{x}_n) \tag{10}$$

for some chosen kernel function $k(\cdot, \cdot)$. The negative sign in the last term is pure convention. Differentiation of Eq. (9) w.r.t. $\boldsymbol{\alpha}$ thus yields the solution

$$\boldsymbol{\alpha} = K^{-1} \cdot (\boldsymbol{y} - \lambda \mathbf{1}) \ , \tag{11}$$

under the assumption that K is invertible.[2] We can solve for the Lagrange multiplier λ by using

$$\mathbf{1} \cdot \boldsymbol{\alpha} = 0 = \mathbf{1}^T \cdot K^{-1} \cdot (\boldsymbol{y} - \lambda \mathbf{1}) \ , \tag{12}$$

which yields

$$\lambda = \frac{\mathbf{1}^T \cdot K^{-1} \cdot \boldsymbol{y}}{\mathbf{1}^T \cdot K^{-1} \cdot \mathbf{1}} \ . \tag{13}$$

As the last step, we need to insert our results back into Eq. (1). Since $y_n = \boldsymbol{w} \cdot \boldsymbol{x}_n + b$ must hold for any n, we can get the bias $b = y_1 - \boldsymbol{w} \cdot \boldsymbol{x}_1$. We thus obtain

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + y_1 - \boldsymbol{w} \cdot \boldsymbol{x}_1 = y_1 + \boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_1) \tag{14}$$

$$= y_1 + \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \cdot (\boldsymbol{x} - \boldsymbol{x}_1) \ . \tag{15}$$

Alternatively, if we use a kernel function $k(\cdot, \cdot)$, this reads

$$f(\boldsymbol{x}) = y_1 + \sum_{n=1}^{N} \alpha_n \left(k(\boldsymbol{x}_n, \boldsymbol{x}) - k(\boldsymbol{x}_n, \boldsymbol{x}_1) \right) \ , \tag{16}$$

which is our final result.

[2] In practical experiments we noticed that K is usually invertible if and only if we choose a non-trivial kernel function.

3 Demonstration with toy data

In order to demonstrate support vector interpolation in practice, we use the following toy example: We take the Hermitian polynomial of fourth order,

$$H_4(x) = 16x^4 - 48x^2 + 12\,, \tag{17}$$

and draw data from it on a regular grid from $x = -2$ to $x = 2$ in steps of $\Delta x = 0.2$. As kernel function, we choose the Cauchy kernel

$$k(x_1, x_2) = \frac{1}{1 + (\frac{x_1 - x_2}{\sigma})^2}\,, \tag{18}$$

where $\sigma > 0$ is the kernel width. Figure 1 shows the resulting interpolation function and how its appearance depends on the choice of σ.

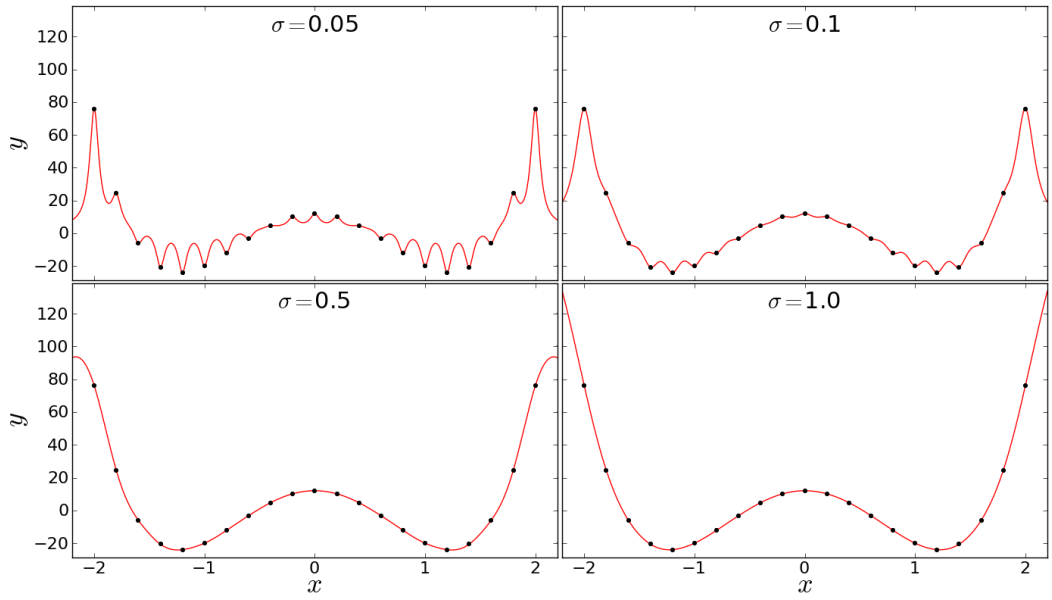

FIGURE 1: Support vector interpolation on toy data drawn from 4-th order Hermite polynomial using different kernel width parameters σ.

The kernel width σ is the single only hyperparameter of support vector interpolation. This makes optimisation of the SVI model almost trivial, e.g., using a one-dimensional brute-force grid search. In practice, we would find the best value of σ using cross-validation on the given training data set.

4 Interpolation of extinction A_0

In the previous section, we demonstrated the performance of SVI with toy data. We now want to turn to the interpolation of actual BP/RP spectra in order to assess how reliable SVI is in

TABLE 1: Leave-one-out cross-validation RMS errors on log-flux for SVI and linear interpolation (on log-flux and linear flux).

pixel	SVI on log-flux	linear interpolation on log-flux	linear interpolation on flux
20	0.0748	0.0025	0.0247
30	0.0856	0.0016	0.0311
40	0.0550	0.0024	0.0112
80	0.0153	0.0024	0.0017
90	0.0061	0.0025	0.0021
100	0.0257	0.0025	0.0013

comparison to the currently implemented linear interpolation. For this purpose, we investigate the interpolation of BP/RP pixel flux as a function of A_0. The other parameters $T_{\rm eff}$, $\log g$ and [Fe/H] are fixed in this setting.

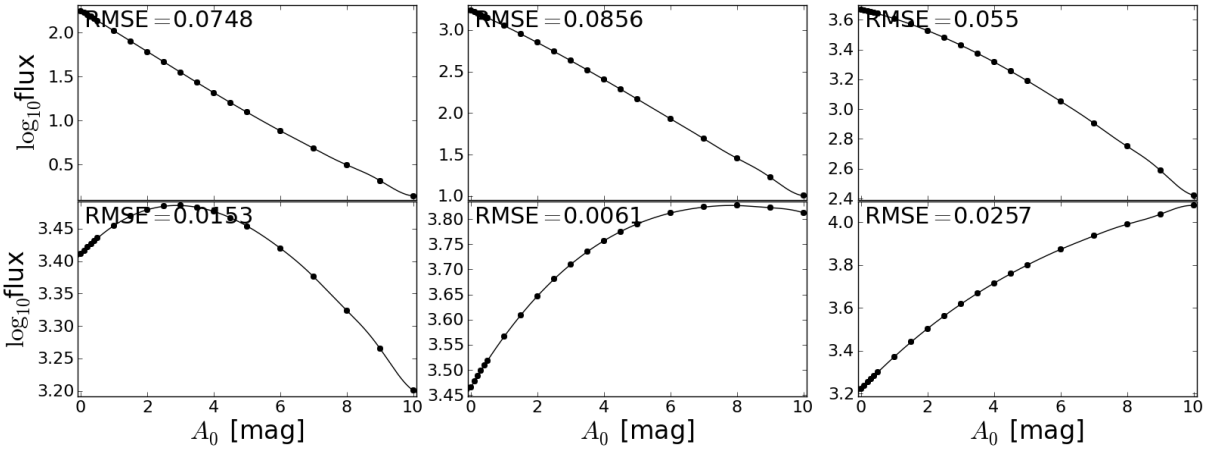

FIGURE 2: Results of SVI for fluxes in different pixels as a function of A_0. In all cases, the star had $T_{\rm eff} = 5\ 800$K, $\log g = 4.5$dex and [Fe/H]$= 0$dex. The panels are pixels 20, 30, 40 (top row), i.e., BP, and 80, 90, 100 (bottom row), i.e., RP. While the panels show the interpolant when trained on all given data points, the numbers quote leave-one-out cross-validation RMS error in log-flux.

Evidently, the kernel width σ needs to be fine-tuned very carefully. To this end, we optimised σ on a very fine brute-force grid, minimising the leave-one-out cross-validation error on the given data. Figure 2 shows the SVI results for six different pixels (20, 30, 40 in BP and 80, 90, 100 in RP). The results look overall good. However, in Fig. 2 we only establish that, first, the interpolant indeed goes through all given training points, and second, the interpolant looks reasonably smooth.

In order to get a more detailed assessment, we compare the RMS errors of SVI to those of linear interpolation. The difference to Fig. 2 is that we now take one point out of the training

set, train the SVI on the remaining data, and then predict the removed point (leave-one-out cross validation). As is clearly evident from Table 1, SVI has the *worst* performance in all examples. SVI on log-flux has larger errors than linear interpolation on log-flux. It has also larger errors than linear interpolation on linear flux (whose errors have been assessed in log-flux). Since the latter is the current implementation of the GSP-Phot forward model, SVI offers no benefit. We also tested kernel functions other than the Cauchy kernel, namely the Gaussian kernel and the linear-spline kernel. However, none of these lead to a significant change of our conclusion. Indeed, the linear-spline kernel significantly reduced the SVI errors but they were still not competitive with linear interpolation.

5 Conclusions

We conclude that support vector interpolation does work but, at the same time, does not provide better results than the currently implemented linear interpolation. Certainly, SVI cannot help us to reduce the size of the forward model grid, i.e., it cannot help us to overcome the GSP-Phot memory problem. Therefore, there is no reason to further pursue this idea.

References

[RAN-024], Andrae, R., 2015, *Least-squares Support Vector Regression for GSP-Phot*, GAIA-C8-TN-MPIA-RAN-024, URL http://www.rssd.esa.int/cs/livelink/open/3313106

[RAN-025], Andrae, R., 2015, *Software Release Note cycle 18 (for OR5 stage III)*, GAIA-C8-TN-MPIA-RAN-025

[RAN-027], Andrae, R., 2015, *Resolving the GSP-Phot outlier problem*, GAIA-C8-TN-MPIA-RAN-027, URL http://www.rssd.esa.int/cs/livelink/open/3364354

Deng, N., Tian, Y., Zhang, C., 2012, *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*, Chapman & Hall/CRC, 1st edn.