Fast, Robust Classification with Applications to Periodic Variable Stars

Siqi Wu, Mark Rogers, and James Long

May 10, 2012

1 Introduction

In this work we design an efficient, robust support vector machine (SVM) algorithm by combining 1) automated tuning of the regularization parameter λ and 2) classification of interval data. The performance of this SVM is tested using periodic variable star data where features are believed to lie within confidence intervals. The paper begins with a motivating example – classification of astronomical time series (Section 2), continues with a discussion of interval SVMs and efficient path algorithms for these models (Sections 3 and 4), and concludes with an application of interval SVMs to two data sets from astronomy (Section 5).

2 The Data

2.1 Background on Periodic Variables

Modern astronomical surveys observe millions of light sources (stars, galaxies, asteroids) over the course of a mission lasting a few years. Periodic variables, sources which vary periodically in brightness over time, are some of the most interesting. In the 1920's, periodic variables were crucial in Edwin Hubble's discovery the existence of galaxies [2]. More recently, periodic variables have played an important role in determining expansion of the universe [8].

Periodic variables may be divided into a few dozen classes based on physical properties of light sources. Separating the sources into classes is a critical step in turning raw astronomical observations into scientific knowledge. The size of modern data sets requires that much of this work be automated by machine learning and statistical classifiers.

Figure 1 displays the light curve (i.e. time series) of a periodic variable belonging to the class Classical Cepheid. The points in the top plot represent flux measurements in magnitudes (i.e. brightness of the source) made by the telescope at particular times. The 0 point on the time axis is arbitrary. Using Fourier methods, one can estimate a period using these measurements. The lower plot of Figure 1 displays the flux measurements of the same object. However here the x-axis is phase of each time measurement, computed using the estimated period of 4.51 days. Here we can observe the structure of the periodic variation. This is known as the *folded light curve*.

Figure 2 displays an example of a Mira light curve. From the y-axis we can see that this source has higher amplitude than the Classical Cepheid (this is typical of the Mira class) and more sinusoidal variation (also typical). Note that the Fourier methods appear to have estimated an incorrect period for this source. The true period appears to be around 161 days, half of the estimate.

2.2 Classification Methodology

There has been a lot of recent progress in the astronomy literature towards developing highly accurate classifiers for the time series generated by periodic variables [4, 10, 6]. The standard approach works as follows. A telescope observes a source j at times $t_1, \ldots, t_l$, recording flux measurements (how bright the object is) of $m_1, \ldots, m_l$. Typically there are measurements of uncertainty on the flux measurements

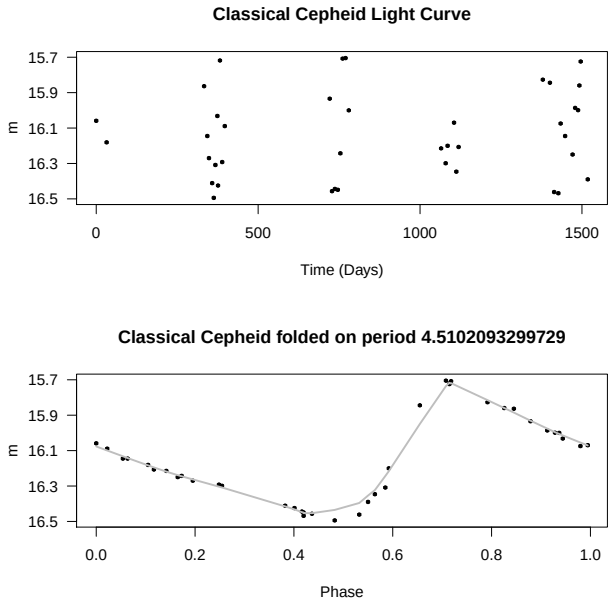

Figure 1: Light curve of a Classical Cepheid type variable star. The brightness of the star is on the y-axis. For the top plot the brightness is plotted against time. This time series is periodic. Using Fourier methods we can estimate a period of 4.51 days. We can convert the times from the top plot into phase ((time modulo period) / period)). Brightness versus phase is plotted in the bottom plot. Here we can observe the structure in the time series, which is usually similar for stars of the same class but different for stars of different classes.

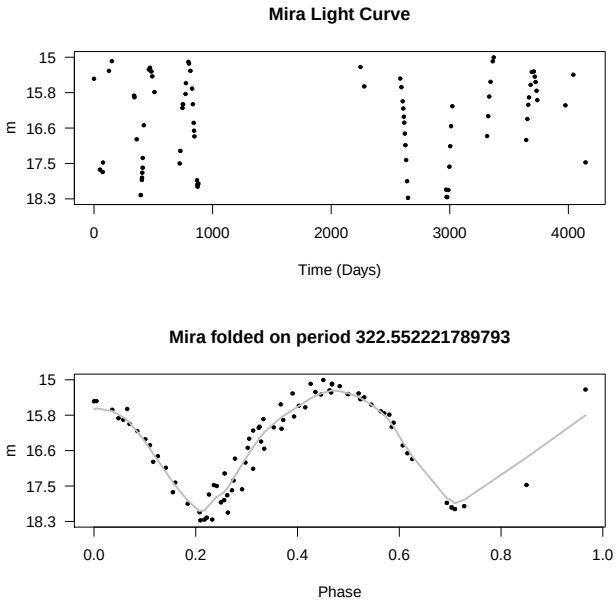

Figure 2: Light curve of a Mira type variable star. Mira variable stars have higher amplitude and longer periods than Cepheid variables. Notice that the period estimate of 322.55 appears to be incorrect. The true period is likely half of this.

$e_1, \ldots, e_l$. So each source j is initially characterized by an l tuples $D_j = \{(t_i, m_i, e_i)\}_{i=1}^{l}$. Note that l, the number of times the source is observed, is different for each D_j. Also the time sampling of the flux measurements is irregular and different for each source. Associated with each source is a classification, such as Mira, Classical Cepheid, RR Lyrae, etc.

In order to construct a classifier, features are *extracted* from D_j (i.e. take functions of D_j) that will separate sources into different classes. Features vary from study to study, but typical ones include period (inverse of strongest Fourier frequency), amplitude, skew, and estimates of derivatives.

If we compute p features and have a total of n training stars of known class $(D_1, \ldots D_n)$, then we can use standard classification techniques on this $n \times p$ data matrix to construct a classifier. Given this classifier, we can then assign a class to a new source by extracting features and running the features through the classifier.

2.3 Uncertain Features

A major problem with the standard approach is the high levels and heteroskedastic nature of the uncertainty in the features due to having poorly sampled time series (l is small), high error in the flux measurements (e_i are systematically large), or irregular nature of sampling (t_i's are highly concentrated, giving a poor representation of variation). This leads to situations where features meant to represent physical quantities of the time series, such as amplitude or period, are incorrect. As an example in Figure 3 we plot the

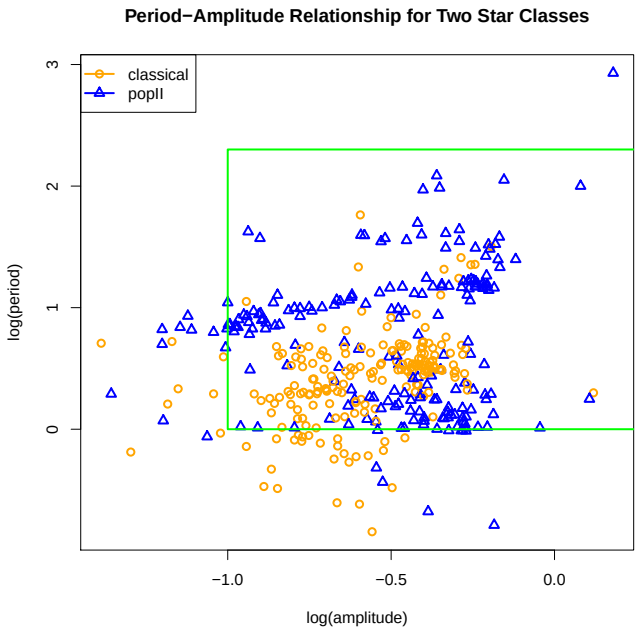

Figure 3: Period-Amplitude relationship for two classes: *Classical Cepheid* stars and *Population II Cepheid* stars. The green box represents (roughly) the physically possible range of periods and amplitudes. Clearly some of these features have been estimated incorrectly.

period and amplitude features for two classes of stars with a box around known physical limits for these features for these two classes. Some of the observations have values outside these physical limits, clearly indicating that the features are incorrectly estimated. This suggests that classifiers which incorporate feature uncertainty may be able to achieve improved performance.

3 SVMs for Interval Data

Support Vector Machines (SVMs) are a popular method for constructing classifiers. Let $x_i \in \mathbb{R}^p$ be the vector of features for observation i and y_i its class (either -1 or $+1$). We determine the SVM classifier by solving the optimization problem,

$$\min_{\beta,\beta_0} \sum_{i=1}^{n} (1 - y_i(\beta^T x + \beta_0))_+ + \frac{\lambda}{2}\beta^T\beta \tag{1}$$

The optimal β^* and β_0^* are used to classify a new observation x using $sign(\beta^{*T}x + \beta_0)$. λ is a tuning parameter that balances a tradeoff between maximizing the margin and separating observations in different classes. An optimal λ may be chosen using 0-1 loss on a test set or through cross-validation. See [12] for an extensive overview of SVMs.

Various extensions to the standard SVM have been proposed that seek to incorporate uncertainty in the features into the optimization problem. These formulations can often be written as minimizing the objective function for a worst case allocation of features in some fixed set, i.e.

$$\min_{\beta,\beta_0} \max_{z \in \mathcal{X}(\rho)} \sum_{i=1}^{n} (1 - y_i(\beta^T z_i + \beta_0))_+ + \frac{\lambda}{2}\beta^T\beta \tag{2}$$

where ρ is some scalar and $\mathcal{X}(\rho)$ defines some set to which $(z_1, \ldots, z_n)$ belong. See [3, 13, 1] for discussions of some of these models.

One way to represent feature uncertainty is to specify that observations must lie within hyper-rectangles of feature space. Formally we let X be an $n \times p$ matrix where row i represents the center of the hyper-rectangle for observation j. We define Σ to be an $n \times p$ matrix where $\Sigma_{i,j}$ is the relative uncertainty on observation i, on feature j. Following the notation of the SVM above we can express this uncertainty model as

$$\mathcal{X}(\rho) = \{Z : X - p\Sigma \leq Z \leq X + \rho\Sigma\} \tag{3}$$

ρ is a global uncertainty parameter that shrinks or expands the hyper-rectangles uniformly. We could think of ρ as a robustness parameter which larger ρ leads to more robust perturbations of the input data. Or from a probabilistic stand–point, larger ρ will increase the chance that a random feature vector is in $\mathcal{X}(\rho)$, thus ensuring separation with higher probability.

This model, but without the $\frac{\lambda}{2}\beta^T\beta$ regularization, was considered in El Ghaoui [7]. It is easy to show that for this form of $\mathcal{X}(\rho)$ one can explicitly solve for the inner max in equation (2) and obtain

$$\min_{\beta} \sum_{i=1}^{n} (1 - y_i(\beta_0 + \beta^T x_i) + \rho\sigma_i^T|\beta|)_+ + \frac{\lambda}{2}||\beta||_2^2. \tag{4}$$

where σ_i is the i^{th} row of Σ. In [7], El Ghaoui drew similarities between the $\rho\sigma_i^T|\beta|$ in this expression and standard L_1 regularization for SVMs. For example, if $\Sigma_{i,j} = 1\ \forall i, j$ then then expression becomes $\rho||\beta||_1$. This interpretation of $\mathcal{X}(\rho)$ in part motivates the use of ρ as a tuning parameter rather than a quantity that is fixed by the inherent uncertainty in the data.

In this work we will fit model (4) to the periodic variable star data sets. This provides an opportunity to make use of known feature uncertainty in our classifier. In addition, the connections between ρ and sparsity, may allow us to obtain highly interpretatable classifiers, with few non-zero feature coefficients.

Before we apply the model to the data, we describe a path algorithm that allows us to fit the model for all values of λ, rather than performing a brute force grid search. This is an adaptation of Hastie's path algorithm for standard SVMs [9].

4 Path Algorithm for Interval SVM

4.1 Problem formulation

Suppose n data points are given: (x_i, y_i) for $i = 1, ..., n$. Let $\mathcal{I}_+ = \{i : y_i = +1\}$ and $\mathcal{I}_- = \{i : y_i = -1\}$ be the set of positive class and negative class respectively. Also let $n_+ = |\mathcal{I}_+|$ and $n_- = |\mathcal{I}_-|$ be the number of data points in these two classes. In this section, we will derive an efficient algorithm to compute the entire regularization path for the interval SVM problem (4). Introducing slack variables, the problem becomes:

$$\begin{aligned} \min & \sum_{i=1}^{n} \xi_i + \lambda\frac{\beta^T\beta}{2}, \\ \text{subject to } & \xi_i \geq 1 - y_i(\beta_0 + \beta^T x_i) + \rho\sigma_i^T t, \\ & \xi_i \geq 0, \text{ for } i = 1, 2, ..., n; \\ & -t_j \leq \beta_j \leq t_j, \\ & t_i \geq 0, \text{ for } j = 1, 2, ..., p. \end{aligned} \tag{5}$$

Let the above problem be the primal problem. The Lagrangian for this problem is

$$\begin{aligned} L(\xi, \beta_0, \beta, t, \alpha, \gamma, \nu, \mu, c) = \ & \textstyle\sum_{i=1}^{n}(1 - \alpha_i - \gamma_i)\xi_i + \lambda\frac{\beta^T\beta}{2} - (\sum_{i=1}^{n} \alpha_i y_i x_i - (\mu - \nu))^T\beta \\ & \textstyle - \sum_{i=1}^{n} \alpha_i y_i \beta_0 + \sum_{j=1}^{p}(\rho\alpha_i\sigma_{ji} - \mu_j - \nu_j - c_j)t_j. \end{aligned}$$

Minimizing with respect to the primal variables (ξ, β_0, β, t), we derive the dual problem as follows:

$$\max_{\alpha,\mu,\nu} \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda} || \sum_{i=1}^{n} \alpha_i y_i x_i - (\mu - \nu)||_2^2, \tag{6}$$
$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0,\ \rho \sum_{i=1}^{n} \alpha_i \sigma_i \geq \mu + \nu,$$
$$\alpha \in [0,1],\ \mu \geq 0,\ \nu \geq 0.$$

One can easily determine the KKT conditions of the interval SVM problem:
(1) Primal stationarity:

$$\frac{\partial}{\partial \beta} : \beta = \frac{1}{\lambda}(\sum_{i=1}^{n} \alpha_i y_i x_i + \nu - \mu) \tag{7}$$
$$\frac{\partial}{\partial \beta_0} : \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{8}$$
$$\frac{\partial}{\partial \xi} : 1 - \alpha - \gamma = 0. \tag{9}$$

(2) Complementary slackness:

$$\alpha_i(1 - y_i(\beta_0 + \beta^T x_i) + \rho \sigma_i^T t - \xi_i) = 0 \tag{10}$$
$$\gamma_i \xi_i = 0 \tag{11}$$
$$\mu_j(t_j - \beta_j) = 0 \tag{12}$$
$$\nu_j(t_j + \beta_j) = 0 \tag{13}$$
$$c_j t_j = 0 \tag{14}$$
$$\rho \sum_{i=1}^{n} \alpha_i \sigma_{ji} - (\mu_j + \nu_j + c_j) = 0 \tag{15}$$

(3) Primal feasibility and; (4) Dual feasibility. Both omitted here due to triviality.

From the above optimality conditions, it is easy to observe the following statements hold:
(i) If $\xi_i > 0$, then $\gamma_i = 0$ and so $\alpha_i = 1$. Therefore by (7) we have $y_i(\beta_0 + \beta^T x_i) < 1 + \rho \sigma_i^T t$, which indicates that the data point i is on the left of the elbow of the hinge loss.
(ii) If $y_i(\beta_0 + \beta^T x_i) > 1 + \rho \sigma_i^T t$, that is, when the i-th data point is on the right of the elbow, then $\xi = 0$ and by equation (6) and (8) $\alpha_i = 0$.
(iii) If $y_i(\beta_0 + \beta^T x_i) = 1 + \rho \sigma_i^T t$, that data point is on the elbow. In this case we can only know $\alpha_i \in [0,1]$.
(iv) It is obvious that at optimal, we have $t = |\beta|$. Hence if $\beta_j > 0$ we have $t_j > 0$ and so $c_j = 0$. On the other hand, since $t_j + \beta_j > 0$, $\nu_j = 0$. Therefore in this case, $\mu_j = \rho \sum_{i=1}^{n} \alpha_i \sigma_{ji}$.
(v) Similarly, if $\beta_j < 0$, $c_j = 0$, $\mu_j = 0$ and $\nu_j = \rho \sum_{i=1}^{n} \alpha_i \sigma_{ji}$.

Now for convenience let $f(x) = \beta_0 + \beta^T x$ and define the following index sets for the data points:

$$\mathcal{E} = \{i : y_i f(x_i) = 1 + \rho \sigma_i^T |\beta|, 0 \leq \alpha_i \leq 1\},\ \mathcal{E} \text{ for Elbow},$$
$$\mathcal{L} = \{i : y_i f(x_i) < 1 + \rho \sigma_i^T |\beta|, \alpha_i = 1\},\ \mathcal{L} \text{ for Left of the elbow},$$
$$\mathcal{R} = \{i : y_i f(x_i) > 1 + \rho \sigma_i^T |\beta|, \alpha_i = 0\},\ \mathcal{R} \text{ for Right of the elbow}.$$

Note that the definition of the above sets are similar to those defined in [9] except that we have an extra term $\rho \sigma_i^T |\beta|$ that is related to the uncertainty of the data. In addition, since the absolute value of β is

involved in our formulation, we also need to keep track of the following index sets for the parameter vector β:

$$\mathcal{V}_+ = \{j : \beta_j > 0, \nu_j = 0\},$$
$$\mathcal{V}_- = \{j : \beta_j < 0, \mu_j = 0\},$$
$$\mathcal{Z} = \{j : \beta_j = 0, \sum_{i=1}^{n} \alpha_i \sigma_{ji} \geq \mu_j + \nu_j\}.$$

4.2 Initialization

We will determine the initial state of the parameters and the sets defined above. Without loss of generality, we assume that $n_+ \geq n_- > 0$. We start with $\lambda = +\infty$ or equivalently $C = 0$. In this case, it is easy to see that $\beta = 0$.

Lemma 4.1. *Given $n_+ \geq n_- > 0$ and $\beta = 0$, the optimal value for the initial value of β_0 is $\beta_0 = 1$ and the loss is $\sum_{i=1}^{n} \xi_i = 2n_-$. If $n_+ > n_-$, then $\beta_0 = 1$.*

Proof. With $\beta = 0$, the primal problem becomes:

$$\begin{array}{rl} \min & \sum_{i=1}^{n} \xi_i, \\ \text{subject to} & \xi_i \geq 0,\ \xi_i \geq 1 - y_i \beta_0. \end{array}$$

Hence for $i \in \mathcal{I}_+$, $\xi_i \geq 1 - \beta_0$ and for $i \in \mathcal{I}_-$, $\xi_i \geq 1 + \beta_0$. At optimal, the equal sign must hold since we are minimizing the sum $\sum_{i=1}^{n} \xi_i$. Also, ξ_i's are nonnegative, therefore we have $1 - \beta_0 \geq 0$ and $1 + \beta_0 \geq 0$ and so $\beta_0 \in [-1, 1]$. Furthermore, the objective function can be written in terms of n_+ and n_-:

$$\sum_{i=1}^{n} \xi_i = n_+(1 - \beta_0) + n_-(1 + \beta_0) = (n_- - n_+)\beta_0 + (n_+ + n_-),$$

which is a linear function in β_0. If $n_- = n_+$, then the above function is simply a constant $2n_-$ and so β_0 can be arbitrary chosen in $[-1, 1]$; on the other hand if $n_+ > n_-$, the linear function is decreasing in β_0 and so to minimize the sum, we can pick β_0 on the right end point of $[-1, 1]$. So $\beta_0 = 1$ if $n_+ > n_-$.

From now on, we can safely assume that $n_+ > n_-$, in which case the initial state is $\beta = 0$, $\beta_0 = 1$, $\xi_i = 0$ for $i \in \mathcal{I}_+$ and $\xi_i = 2$ for $i \in \mathcal{I}_-$. Thus by optimality conditions, for any $i \in \mathcal{I}_-$, $\gamma_i = 0$ and so $\alpha_i = 1$. In addition, since $0 = \sum_{i=1}^{n} \alpha_i y_i = \sum_{i \in \mathcal{I}_+} \alpha_i - \sum_{i \in \mathcal{I}_-} \alpha_i$, we have $\sum_{i \in \mathcal{I}_+} \alpha_i = \sum_{i \in \mathcal{I}_-} \alpha_i = n_-$. The following lemma determines the initial value for the dual valuables:

Lemma 4.2. *Let $\tilde{\beta} = \sum_{i=1}^{n} \alpha_i y_i x_i + \nu - \mu$,*

$$\begin{array}{rl} (\alpha^*, \mu^*, \nu^*) = & \arg\min ||\tilde{\beta}||_2^2, \\ & \textit{subject to: } 0 \leq \alpha_i \leq 1, \forall i \in \mathcal{I}_+; \alpha_i = 1, \forall i \in \mathcal{I}_-; \\ & \sum_{i \in \mathcal{I}_+} \alpha_i = n_-, \rho\Sigma\alpha \geq \mu + \nu, \mu \geq 0, \nu \geq 0. \end{array}$$

and $c^* = \rho\Sigma\alpha^* - \mu^* - \nu^*$. Then for some λ_0 we have for all $\lambda > \lambda_0$, $(\alpha(\lambda), \mu(\lambda), \nu(\lambda), c(\lambda)) = (\alpha^*, \mu^*, \nu^*, c^*)$.

Proof. The proof follows directly from Lemma 2 in [9]. The key observation is that $\sum_{i=1}^{n} \alpha_i = 2n_-$ and this sum will remain constant for a while as β grows from zero. Then we can plug this sum into the dual problem (6) and solve it with additional constraints: $\alpha_i = 1, \forall i \in \mathcal{I}_-$ and $\sum_{i \in \mathcal{I}_+} \alpha_i = n_-$.

Now let $\beta^* = \sum_{i=1}^n \alpha_i^* y_i x_i + \nu^* - \mu^*$ be the fixed coefficient direction corresponding to the initial (α^*, μ^*, ν^*). By the above lemma and the optimality conditions, we have for all $\lambda > \lambda_0$:

$$\beta(\lambda) = \frac{\sum_{i=1}^n \alpha_i^* y_i x_i + \nu^* - \mu^*}{\lambda}.$$

We now need to determine the point λ_0 that the dual variables (α, μ, ν, c) start to change. At the very beginning when $\lambda = +\infty$, there are two possible scenarios:

- There exist at least two data points in $\mathcal{I}_+$ with $0 < \alpha_i^* < 1$. Notice that there cannot be only one such element due to the integer constraint $\sum_{i \in \mathcal{I}_+} \alpha_i^* = n_-$.
- For all $i \in \mathcal{I}_+$, α_i^* is either 0 or 1.

For the first scenario, arbitrary pick an $i_+ \in \mathcal{I}_+$ such that $0 < \alpha_{i_+}^* < 1$. Since any of these points will stay in the elbow until a point in $\mathcal{I}_-$ enters the elbow, we can consider the element in $\mathcal{I}_-$ that first reaches its margin, i.e. $i_- = \arg\min_{i \in \mathcal{I}_-} x_i^T \beta^* + \rho \sigma_i^T |\beta^*|$. Therefore at $\lambda = \lambda_0$, as both i_+ and i_- are at their respective margin, by the definition of elbow, we have the following system of equations:

$$\begin{cases} \beta_0 + \frac{1}{\lambda} x_{i_+}^T \beta^* = 1 + \frac{1}{\lambda} \rho \sigma_{i_+}^T |\beta^*|, \\ \beta_0 + \frac{1}{\lambda} x_{i_-}^T \beta^* = -1 - \frac{1}{\lambda} \rho \sigma_{i_-}^T |\beta^*|. \end{cases}$$

Solving for λ and β_0 yields:

$$\begin{cases} \lambda_0 = \frac{1}{2}(\beta^{*T}(x_{i_+} - x_{i_-}) - \rho |\beta^*|^T (\sigma_{i_+} + \sigma_{i_-})), \\ \beta_0 = \frac{-\beta^{*T}(x_{i_+} + x_{i_-}) + \rho |\beta^*|^T (\sigma_{i_+} - \sigma_{i_-})}{\beta^{*T}(x_{i_+} - x_{i_-}) - \rho |\beta^*|^T (\sigma_{i_+} + \sigma_{i_-})}. \end{cases} \tag{16}$$

For the second scenario, for the initial parameter to change, a point in $\mathcal{I}_-$ and a point in $\mathcal{I}_+$ with $\alpha_i^* = 1$ must reach the margin simultaneously. Therefore we can let $i_+ = \arg\max_{i \in \mathcal{I}_+, \alpha_i = 1} x_i^T \beta^* - \rho \sigma_i^T |\beta^*|$ and obtain λ_0 and β_0 by solving the above equations.

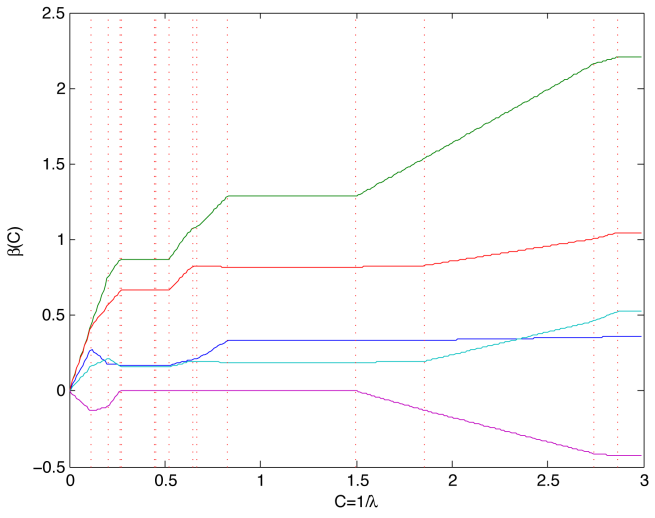

Figure 4: The path of β for the simulated data.

4.3 The Path

As in [9], our algorithm keeps track of the following events:

1. One or more points from $\mathcal{L}$ have just entered $\mathcal{E}$;
2. One or more points from $\mathcal{R}$ have just reentered $\mathcal{E}$;
3. One or more points in $\mathcal{E}$ have just left the set, to join either $\mathcal{R}$ or $\mathcal{L}$.

In addition, we also need to keep track of the following events involving the sign of the components of β:

4. One or more indices of the parameter β enter $\mathcal{Z}$, that is, β_j becomes zero, initially being either positive or negative;
5. One or more indices of the parameter β enter $\mathcal{V}_+$ from $\mathcal{Z}$;
6. One or more indices of the parameter β enter $\mathcal{V}_-$ from $\mathcal{Z}$.

4.4 Piecewise Linearity between Events

By continuity, all of the index sets defined will stay the same until the next event occurs. Index by the superscript l the sets immediately after the l-th event occurs. Likewise, index all the parameters in the same manner, and let f^l be the classify function at this point. Also, for notation consistency define $\alpha_0 = \lambda\beta_0$ and so $\alpha_0^l = \lambda^l\beta_0^l$. Consider for $\lambda^l > \lambda > \lambda^{l+1}$, we have:

$$\begin{aligned} f(x) &= [f(x) - \frac{\lambda^l}{\lambda}f^l(x)] + \frac{\lambda^l}{\lambda}f^l(x) \\ &= \frac{1}{\lambda}[x^T\sum_{i\in\mathcal{E}^l}(\alpha_i - \alpha_i^l)y_ix_i + x^T(\nu - \nu^l) - x^T(\mu - \mu^l) + (\alpha_0 - \alpha_0^l) + \lambda^l f^l(x)], \end{aligned} \tag{17}$$

where the last equality follows from the fact that $\{1, ..., n\} = \mathcal{L} \cup \mathcal{E} \cup \mathcal{R}$, $\alpha_i = 1$ on $\mathcal{L}$ and $\alpha_i = 0$ on $\mathcal{R}$. The value of α_i for i not in the elbow remains unchanged between events. Since each data point in $\mathcal{E}^l$ is to stay in $\mathcal{E}$ for $\lambda \in (\lambda^{l+1}, \lambda^l)$, we have:

$$y_jf(x_j) = 1 + \rho\sigma_j^T|\beta|, \forall j \in \mathcal{E}.$$

Plug in (17) for $f(x_j)$, the above equation becomes:

$$\frac{1}{\lambda}[\sum_{i\in\mathcal{E}^l}(\alpha_i - \alpha_i^l)y_jy_ix_j^Tx_i + y_jx_j^T(\nu - \mu - (\nu^l - \mu^l)) + y_j(\alpha_0 - \alpha_0^l) + \lambda^l(1 + \rho\sigma_j^T|\beta^l|)] = 1 + \rho\sigma_j^T|\beta|. \tag{18}$$

We can expression the parameter β_i in terms of α. For any $i \in \mathcal{V}_+^l$, since $\nu_i = 0$, $\mu_i = \rho\sum_{k=1}^n \alpha_k\sigma_{ik}$. Plug all these parameters into the formula for β_i, we have

$$\beta_i = \frac{1}{\lambda}\sum_{k=1}^n(\alpha_iy_kx_{ik} - \mu_i) = \frac{1}{\lambda}\sum_{k=1}^n\alpha_k(y_kx_{ik} - \rho\sigma_{ik}) = \frac{1}{\lambda}\sum_{k=1}^n\alpha_kw_{ik},$$

where $w_{ik} = y_kx_{ik} - \rho\sigma_{ik}$. Similarly, for $i \in \mathcal{V}_-$, since $\mu_i = 0$, $\nu_i = \rho\sum_{k=1}^n \alpha_k\sigma_{ik}$ and so

$$\beta_i = \frac{1}{\lambda}\sum_{k=1}^n(\alpha_iy_kx_{ik} + \nu_i) = \frac{1}{\lambda}\sum_{k=1}^n\alpha_k(y_kx_{ik} + \rho\sigma_{ik}) = \frac{1}{\lambda}\sum_{k=1}^n\alpha_kz_{ik}.$$

Let $h_{jk} = \sum_{i \in \mathcal{V}_+^l} \sigma_{ij} w_{ik} - \sum_{i \in \mathcal{V}_-^l} \sigma_{ij} z_{ik}$. By splitting β into positive and negative parts, we have:

$$\begin{aligned} \sigma_j^T |\beta| &= \textstyle\sum_{i \in \mathcal{V}_+^l} \sigma_{ij} \beta_i - \sum_{i \in \mathcal{V}_-^l} \sigma_{ij} \beta_i \\ &= \frac{1}{\lambda} \textstyle\sum_{k=1}^n \alpha_k (\sum_{i \in \mathcal{V}_+^l} \sigma_{ij} w_{ik} - \sum_{i \in \mathcal{V}_-^l} \sigma_{ij} z_{ik}) \\ &= \frac{1}{\lambda} \textstyle\sum_{k=1}^n \alpha_k h_{jk}. \end{aligned}$$

Next, look at the term $x_j^T(\nu - \mu)$. Notice that:

$$x_j^T(\nu - \mu) = \textstyle\sum_{i \in \mathcal{V}_-^l} x_{ij} (\rho \sum_{k=1}^n \alpha_k \sigma_{ik}) - \sum_{i \in \mathcal{V}_+^l} x_{ij} (\rho \sum_{k=1}^n \alpha_k \sigma_{ik}) + \sum_{i \in Z^l} x_{ij} (\nu_i - \mu_i)$$

Observe that for $i \in Z^l$, $\beta_i = 0$ and so $\sum_{k=1}^n \alpha_k y_k x_{ik} + \nu_i - \mu_i = 0$. Therefore $\nu_i - \mu_i = -\sum_{k=1}^n \alpha_k y_k x_{ik}$. Hence we have:

$$x_j^T(\nu - \mu) = \sum_{k=1}^n \alpha_k (\rho \sum_{i \in \mathcal{V}_-} x_{ij} \sigma_{ik} - \rho \sum_{i \in \mathcal{V}_+} x_{ij} \sigma_{ik} + \sum_{i \in Z} y_k x_{ik}) = \sum_{k=1}^n \alpha_k g_{jk},$$

where $g_{jk} = \rho \sum_{i \in \mathcal{V}_-^l} x_{ij} \sigma_{ik} - \rho \sum_{i \in \mathcal{V}_+^l} x_{ij} \sigma_{ik} + \sum_{i \in Z^l} y_k x_{ik}$. Now, let $\delta_k = \alpha_k^l - \alpha_k$. After some rearrangements, Equation (18) becomes:

$$\sum_{k \in \mathcal{E}} \delta_k (y_j y_k x_j^T x_k + y_j g_{jk} - \rho h_{jk}) + y_j \delta_0 = \lambda^l - \lambda,$$

for all $j \in \mathcal{E}$. Thus following [9], by constructing a matrix $\mathbf{K}$ with entry $K_{jk} = y_j y_k x_j^T x_k + y_j g_{jk} - \rho h_{jk}$, the above system of equations can be written:

$$\mathbf{K}\boldsymbol{\delta} + \delta_0 \mathbf{y}_l = (\lambda^l - \lambda)\mathbf{1},$$

where $\mathbf{y}_l$ is a vector of length m whose entries are y_i's for $i \in \mathcal{E}^l$. In addition, since $\sum_{i=1}^n \alpha_i y_i = 0$, we have $\sum_{k \in \mathcal{E}} y_k \delta_k = 0$ and so

$$\mathbf{y}_l^T \boldsymbol{\delta} = 0.$$

So all together we have $m+1$ unknown variables and $m+1$ linear equations. Now let:

$$\mathbf{A}_l = \begin{pmatrix} 0 & \mathbf{y}_l^T \\ \mathbf{y}_l & \mathbf{K}_l \end{pmatrix}, \ \boldsymbol{\delta}^a = \begin{pmatrix} \delta_0 \\ \boldsymbol{\delta} \end{pmatrix}, \ \mathbf{1}^a = \begin{pmatrix} 0 \\ \mathbf{1} \end{pmatrix}.$$

So the linear system can be written as

$$\mathbf{A}_l \boldsymbol{\delta}^a = (\lambda_l - \lambda)\mathbf{1}^a.$$

Let $\mathbf{b}^a = \mathbf{A}_l^{-1} \mathbf{1}^a$, then for $k = 0$ or $k \in \mathcal{E}^l$, we have

$$\alpha_k = \alpha_k^l - (\lambda^l - \lambda) b_k, \tag{19}$$

which shows linearity of α between two events, i.e. $\lambda^{l+1} < \lambda < \lambda^l$. To evaluate the function value of f at the data point x_j, we have

$$f(x_j) = \frac{\lambda^l}{\lambda} (f^l(x_j) - h^l(x_j)) + h^l(x_j), \tag{20}$$

where $h^l(x) = \sum_{k \in \mathcal{E}_l} b_j (y_k x_j^T x_k + g_{jk}) + b_0$.

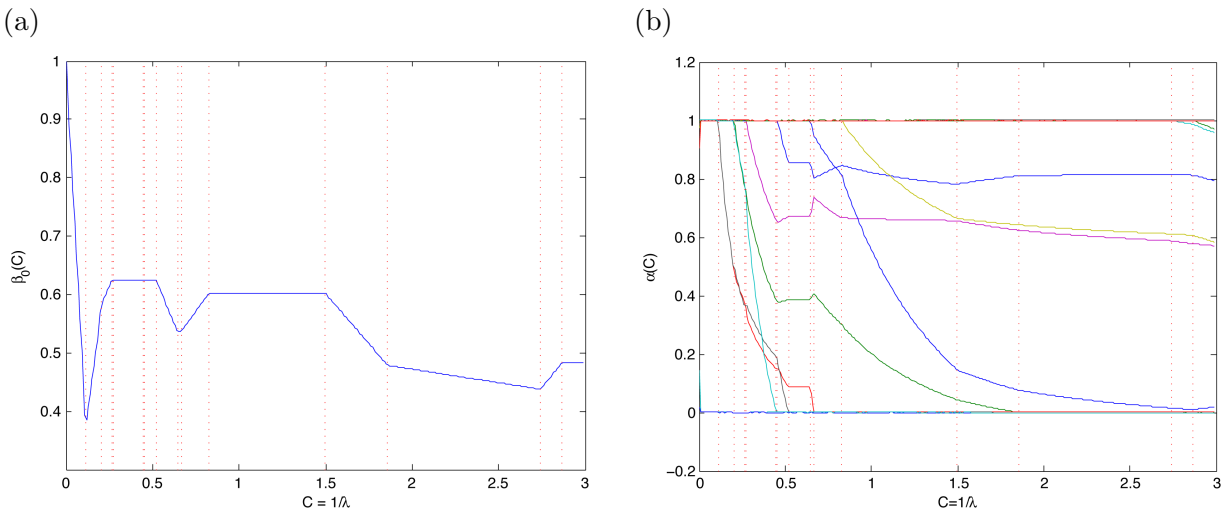

Figure 5: (a) Performance across tuning parameters for RR data set. (b) The path of α for the simulated data.

4.5 Finding the Next Event

The linear path established above continues until one of the following events occur:

1. One of the points (if any) on the elbow $\mathcal{E}^l$ is about to enter either $\mathcal{R}$ ($\alpha_i = 0$) or $\mathcal{L}$ ($\alpha_i = 1$). By solving equation (19) we can obtain a candidate breakpoint λ for the j-th data point to enter the respectively $\mathcal{R}$ and $\mathcal{L}$:
$$\lambda = \frac{\lambda^l b_j - \alpha_j^l}{b_j} \text{ and } \lambda = \frac{\lambda^l b_j - \alpha_j^l + 1}{b_j}.$$

2. One of the points j in either $\mathcal{L}^l$ or $\mathcal{R}^l$ enters the elbow, that is $y_j f(x_j) = 1 + \rho\sigma_j^T|\beta|$. By Equation (20), we have
$$\frac{\lambda^l}{\lambda}(f^l(x_j) - h^l(x_j))y_j + h^l(x_j)y_j = 1 + \frac{1}{\lambda}\sum_{i=1}^{n}\alpha_k h_k.$$
Solving the above equation for λ, we have:
$$\lambda = \frac{\lambda^l(f^l(x_j) - h^l(x_{jk})) - y_j\sum_{k\in\mathcal{L}^l} h_{jk} - y_j\sum_{k\in\mathcal{E}^l}\alpha_k^l h_{jk} + \lambda^l\sum_{k\in\mathcal{E}^l} b_k h_{jk}}{y_j - h^l(x_j) + y_j\sum_{k\in\mathcal{E}^l} b_k h_{jk}}.$$

3. A nonzero component of β becomes zero. This means for each $i \in \mathcal{V}_+^l$, by equation we solve $\sum_{k=1}^n \alpha_k w_{ik} = 0$ and obtain a break point candidate
$$\lambda = \frac{\lambda^l\sum_{k\in\mathcal{E}^l} b_k w_{ik} - \sum_{k\in\mathcal{E}^l}\alpha_k^l w_{ik} - \sum_{k\in\mathcal{L}^l} w_{ik}}{\sum_{k\in\mathcal{E}^l} b_k w_{ik}}.$$
For $i \in \mathcal{V}_-^l$, simply replace w_{ik} by z_{ik}.

4. A zero component of β becomes nonzero. For a component $i \in Z^l$ to become positive, we require $c_i = 0$ and $\nu_i = 0$, therefore $\mu_i = \sum_{k=1}^n \alpha_k y_k x_{ik}$. Together with the optimality conditions, we need to solve $\rho\sum_{k=1}^n \alpha_k\sigma_{ik} = \sum_{k=1}^n \alpha_k y_k x_{ik}$. Same analysis for $i \in Z^l$ to become negative. The formula for λ turns out to have the same form as that in 3.

Compute the all of the above λ's and take the greatest such λ that is smaller than λ^l. That λ will be the break point for the next event.

4.6 Continuation and termination

After finding the λ at which the next event occurs, we can update all the index sets accordingly and continue to find the next event. If the data points are separable in the sense of interval data (that is, there is a hyperplane that separates all the rectangles formed by the data points in the two classes), the algorithm can terminate once $\mathcal{L}^l$ becomes empty. If the points are not separable, then we rely on the user to provide an upper bound or λ or a maximum number of break points.

4.7 A simple simulated example

To test the correctness and efficiency of our proposed algorithm, we construct a test data set as follows: let $n_+ = 10$ and $n_- = 8$, and to keep the example simple let us consider having only $p = 5$ features. For each $i \in \mathcal{V}_+$ and $j = 1, ..., 5$, generate x_{ij} from a standard normal distribution $N(0, 1)$, independently of each other. Similarly, generate each x_{ij} in the negative class from the normal distribution $N(-0.5, 1)$. To construct the precision matrix, for each entry in that matrix, simulate a number from $N(0.1, 0.1)$ and then take the absolute value of it. Also, we set $\rho = 1.0$ and target the range of C to be $[0, 3]$. First, we use a brute-force algorithm to compute the regularization paths for the parameters, making a call to the function that solves one interval SVM problem (`svmInterval`) at a finite break points of along the interval $[0, 3]$. We then run our interval SVM path algorithm on this data set and compare the results of the two approaches. The paths for β are shown in Figure 4 and the plots for β_0 and α can be found in the Appendix. In each of the plot, the paths are computed via the brute-force algorithm and the break points (indicated by the dash lines) are predicted by the path algorithm. It can be seen that our path algorithm correctly identifies all the changing points in this example.

4.8 Computation Complexity

Hastie et al [9] suggests the running time of their algorithm to be $O(n^2m + nm^2)$, where m is the average size of the elbow $\mathcal{E}^l$. Since our algorithm is a variant of theirs (in particular, the core burden of the computation is still to solve the linear system which results in $O(m_l^3)$. Of course, we might have many of such systems the along the path), we conjecture our running time to be the same. We time our path algorithm for fitting the data in the previous section and the elapsed time is 1.01 secs on a 2.66 GHz Intel Core 2 Duo Mac OS machine. As a comparison, the running time for the brute-force algorithm is 66.27 secs for 300 points along the path. Note that the path algorithm gives the true path while the brute-force algorithm is still finding only a few snapshots of the path.

4.9 Matlab and R programs

All the programs for this project are included in the `zip` file submitted together with this hard copy. In particular, the Matlab package `svmIntervalPath` is the implementation of our path algorithm. Also, Matlab and R code for testing and data analysis can be found in that file as well.

5 Application to Variable Star Data Sets

We now return to the periodic variable classification problem and discuss construction of hyper-rectangles, the specific data sets we use, and results from the models.

5.1 Constructing the Intervals from the Time Series

In certain applications, methods for constructing hyper-rectangles (or hyper-spheres) in feature space may be fairly strait forward. For example in El Ghaoui [7], the authors discuss applications to micro-array data where several replicates of some experiment are available, and a hyper-rectangle can be chosen to enclose all replicates.

Constructing Hyper-rectangles from Time Series

1. Order the light curve measurements in time i.e. $\{(t_1, m_1, e_1), (t_2, m_2, e_2), \ldots, (t_l, m_l, e_l)\}$ where $t_i < t_j$ for $i < j$.
2. Slice each light curve into 5 contiguous sections, producing 5 time series. Slice 1 is (t_i, m_i, e_i) for $0 \leq i \leq .5l$, slice 2 for $.125l \leq i \leq .625l$, slice 3 for $.25l \leq i \leq .75l$ ect.
3. Derive features for each of these slices.
4. Put the interval minimum at the lowest value of the feature for the 5 slices. Put the interval max at the maximum feature value across the 5 slices.

Figure 6: Description of interval construction algorithm.

With astronomy data there is no clearly correct way to construct intervals around features which are themselves complicated functions of time series. A few possibilities include:

1. Make width of feature interval proportional to number of measurements in time series. Long time series will have small intervals around features, short time series will have wide intervals. For simple features, such as the standard deviation of the brightness measurements, making the interval width go down at rate $\sqrt{n}$ has a probabilistic interpretation via the central limit theorem.

2. Subsample the time series and derive features for each subsample. Represent observations as hyper-rectangles that contain features derived from every subsample (or a certain fraction of the subsamples).

3. Heuristically put intervals around observations with features that are wrong. For example, the amplitude of a star cannot be greater than 6 magnitudes, so any star with amplitude greater than 6 mags could be given an interval that contains values less than 6.

In this work we take the second approach. While the first approach is attractive for simple features, many features (such as frequency) do not have a $\sqrt{n}$ convergence to the true value. Further, the irregular time sampling means that even simple features may not have a $\sqrt{n}$ convergence. The third approach involves a lot of domain knowledge and will change from application to application. Further, there is no guarantee that features within the range of what is physically possible are actually correct.

Figure 6 describes precisely how we subsample the time series and determine interval widths. The details of this procedure could be changed. For example one could bootstrap sample the time measurements, instead of subsampling contiguous sections. Certain computational considerations enter here. It takes around 1 second to derive features for an average length astronomical time series. So sampling, say 20 times, could become prohibitively expensive if the data set is initially at the limits of what is computationally feasible.

5.2 Preprocessing the data with quantile ranking

Many SVM machine implementation normalize features to ensure that margins are comparable between features e.g. [5]. We normalize our data by assigning normal quantiles to the 2n values (lower and upper bounds of the interval) for each feature. This normalization preserves order relations among intervals in the sense that if one interval is completely above another then after normalization it is completely above, and if there is some overlap before normalization there will be some overlap after. Features containing all zeroes were removed from the dataset and missing values were inferred using the median for the feature.

5.3 The Data Sets

We use light curves from the Optical Gravitational Lensing Survey III (OGLEIII).[1]. Since we want to focus on studying sparsity and the relationship between ρ and C, we choose two small data sets that will be easy to analyze.

1. The CEP dataset contains two classes of Cepheids – Classical Cepheids and Population II Cepheids. Each class consisted of $n = 200$ observations and $p = 58$ features.

2. The RR data set consists of two classes of RR Lyrae stars – RR Lyrae AB and RR Lyrae D, again with $n = 200$ and $p = 58$.

5.4 Results

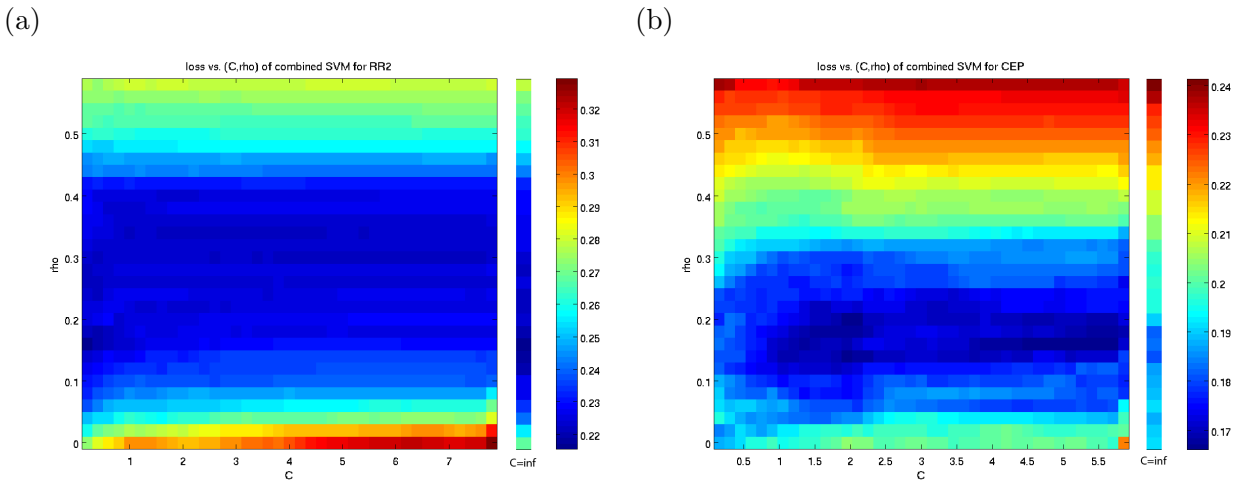

Figure 7: (a) Performance across tuning parameters for RR data set. (b) Performance across tuning parameters for CEP data set.

We treat C and ρ as tuning parameters and grid search across them to find the optimal values. We use a grid search for C, rather than the path algorithm, because it gives more consistent results. This may be due to a bug somewhere in our path code. In order to assess performance on the grid, we perform 10 fold cross validation, with some initial processing to determine where the grid should be placed.

Figure 7 shows cross-validation error rates for given C (x-axis) and ρ y-axis for the RR data set (left) and the CEP data set (right). The color strip on the right shows performance when $C = \infty$, i.e. no L_2 regularization. In both cases, using $\rho > 0$ and $C < \infty$ gives better performance than only relying on one method for regularization. In the left plot (RR data set), there is some evidence of a tradeoff in regularization between ρ and C. By this we mean, as one increases C (regularizes less using the standard penalty), the optimal hyper-rectangle size increases, suggesting this is making up for less L_2 regularization.

One attractive feature of using hyper-rectangles to represent uncertainty is the opportunity to obtain sparse solutions. In Figures 8 (RR) and 9 (CEP) we list the features and coefficients for all features with non-zero coefficients for the models with the lowest error rate. In both cases, over half the features have coefficients exactly 0. It is also true that in both cases many feature have coefficients which are close to, but not quite 0. We might be able to obtain sparser solutions by increasing our ρ. The tradeoff is that this would mean choosing a model that did not obtain optimal misclassification performance.

[1]Light curves can be obtained here: http://ogledb.astrouw.edu.pl/ ogle/CVS/

feature name	beta
percent amplitude	0.83
freq1 harmonics amplitude 1	0.78
freq varrat	0.54
freq signif ratio 21	0.53
std	0.26
freq2 harmonics amplitude 0	0.23
freq1 harmonics freq 0	0.14
beyond1std	0.00
freq1 harmonics amplitude 2	0.00
freq amplitude ratio 21	0.00
amplitude	0.00
skew	0.00
percent difference flux percentile	0.00
freq frequency ratio 21	0.00
freq1 harmonics amplitude 0	0.00
scatter res raw	0.00
stetson j	0.00
qso log chi2 qsonu	0.00
p2p scatter pfold over mad	0.00

Figure 8: RR features with non-zero coefficients. Some coefficient are reported here as 0 due to rounding. Choosing a slightly higher ρ may have caused several of the features with very small coefficients to become exactly 0, leading to a very sparse solution.

5.5 Conclusions

We have developed an algorithm and empirically studied an SVM interval model which incorporates feature uncertainty and produces sparse classifiers. The model improves on classification performance over ignoring feature uncertainty. We feel that the results warrant applying this method to larger and multi-class astronomy data sets. From a data perspective we faced several challenges, the greatest of which was determining intervals for features. This will be common in many high dimensional classification problems where there is uncertainty on features, but no clear way to construct intervals. Generalizing feature extraction methods to produce estimates of uncertainty will make application of interval SVM methods much easier.

A natural extension of the SVM interval concept is to incorporate the feature uncertainty of the unknown test data point into the classifier. Broadly, if a new unlabeled observation has a lot of uncertainty on feature x, then it does not make sense to use feature x heavily in the classifier. However if feature x has small error then a classifier should make as much use of x as possible. Since different observations will have different amounts of noise in different features, it may make sense to develop interval SVM methods that are specifically tuned the the uncertainty of each unlabeled observation. We will work on developing this idea further in future work.

References

[1] A. Ben-Tal, S. Bhadra, C. Bhattacharyya, and J. Saketha Nath. Chance constrained uncertain classification via robust optimization. *Mathematical programming*, 127(1):145–173, 2011.

feature name	**beta**
qso log chi2 qsonu	2.35
beyond1std	1.93
freq1 harmonics amplitude 0	1.41
skew	1.08
freq2 harmonics amplitude 0	1.05
freq1 harmonics freq 0	0.95
freq3 harmonics amplitude 0	0.62
freq1 harmonics rel phase 1	0.44
amplitude	0.33
percent amplitude	0.03
qso log chi2nuNULL chi2nu	0.00
freq amplitude ratio 21	0.00
scatter res raw	0.00
freq amplitude ratio 31	0.00
std	0.00
stetson j	0.00
p2p ssqr diff over var	0.00
median absolute deviation	0.00
flux percentile ratio mid50	0.00
p2p scatter 2praw	0.00
percent difference flux percentile	0.00
freq varrat	0.00
freq3 harmonics amplitude 3	0.00
freq3 harmonics amplitude 1	0.00
max slope	0.00
freq1 harmonics amplitude 1	0.00
flux percentile ratio mid35	0.00
freq frequency ratio 31	0.00
flux percentile ratio mid80	0.00
flux percentile ratio mid65	0.00
freq signif ratio 31	0.00
freq signif ratio 21	0.00

Figure 9: CEP features with non-zero coefficients. Some coefficients are reported as 0 here due to rounding.

[2] R. Berendzen and M. Hoskin. Hubble's announcement of cepheids in spiral nebulae. *Leaflet of the Astronomical Society of the Pacific*, 10:425–440, 1971.

[3] C. Bhattacharyya, LR Grate, M.I. Jordan, L.E. Ghaoui, and I.S. Mian. Robust sparse hyperplane classifiers: application to uncertain molecular profiling data. *Journal of Computational Biology*, 11(6):1073–1089, 2004.

[4] J. Debosscher, LM Sarro, C. Aerts, J. Cuypers, B. Vandenbussche, R. Garrido, and E. Solano. Automated supervised classification of variable stars. *Astronomy and Astrophysics*, 475(3):1159–1183, 2007.

[5] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, A. Weingessel, and M.D. Meyer. Package e1071, 2011.

[6] P. Dubath, L. Rimoldini, M. Süveges, J. Blomme, M. López, LM Sarro, J. De Ridder, J. Cuypers, L. Guy, I. Lecoeur, et al. Random forest automated supervised classification of hipparcos periodic variable stars. *Monthly Notices of the Royal Astronomical Society*, 2011.

[7] L. El Ghaoui, G.R.G. Lanckriet, and G. Natsoulis. *Robust classification with interval data*. Computer Science Division, University of California, 2003.

[8] W.L. Freedman and B.F. Madore. The hubble constant. *Annu. Rev. Astron. Astrophys*, 48:1, 2010.

[9] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *The Journal of Machine Learning Research*, 5:1391–1415, 2004.

[10] J.W. Richards, D.L. Starr, N.R. Butler, J.S. Bloom, J.M. Brewer, A. Crellin-Quick, J. Higgins, R. Kennedy, and M. Rischard. On machine-learned classification of variable stars with sparse and noisy time-series data. *The Astrophysical Journal*, 733:10, 2011.

[11] S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, 35(3):1012–1030, 2007.

[12] B. Schölkopf and A.J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. the MIT Press, 2002.

[13] P.K. Shivaswamy, C. Bhattacharyya, and A.J. Smola. Second order cone programming approaches for handling missing and uncertain data. *The Journal of Machine Learning Research*, 7:1283–1314, 2006.

[14] L. Wang, J. Zhu, and H. Zou. Hybrid huberized support vector machines for microarray classification. In *Proceedings of the 24th international conference on Machine learning*, pages 983–990. ACM, 2007.