DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY


Genetic Algorithm Optimization of Dynamic Support Vector Regression

Citation

Milnes, Thomas Bradford, Christopher Thorpe, and Avi Pfeffer. 2009. Genetic Algorithm Optimization of Dynamic Support Vector Regression. Harvard Computer Science Group Technical Report TR-08-09.

Link

http://nrs.harvard.edu/urn-3:HUL.InstRepos:24825702

Terms of use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material (LAA), as set forth at

https://harvardwiki.atlassian.net/wiki/external/NGY5NDE4ZjgzNTc5NDQzMGIzZWZhMGFlOWI2M2EwYTg

Accessibility

https://accessibility.huit.harvard.edu/digital-accessibility-policy

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story

Genetic Algorithm Optimization of Dynamic Support Vector Regression

Thomas Bradford Milnes
Christopher Thorpe
Avi Pfeffer

TR-08-09

Computer Science Group
Harvard University
Cambridge, Massachusetts

Genetic Algorithm Optimization of Dynamic Support Vector Regression

Thomas Bradford Milnes
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
Email: milnes@post.harvard.edu

Christopher Thorpe
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
Email: cat@eecs.harvard.edu

Avi Pfeffer
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
Email: avi@eecs.harvard.edu

***Abstract*—We show that genetic algorithms (GA) find optimized dynamic support vector machines (DSVMs) more efficiently than the grid search (GS) optimization approach. In addition, we show that GA-DSVMs find extremely low-error solutions for a number of oft-cited benchmarks. Unlike standard support vector machines, DSVMs account for the fact that data further back in a time series are generally less predictive than more-recent data. In order to tune the discounting factors, DSVMs require two new free parameters for a total of five. Because of the five free parameters, traditional GS optimization becomes intractable for even modest grid resolutions. GA optimization finds better results while using fewer computational resources.**

***Index Terms*—support vector machines, genetic algorithms, time series, prediction methods, finance.**

I. Introduction

Traditional support vector regression treats all data points in a time series as equally relevant. [2] presents an extension of these models that allows for the discounting of older, less-relevant data points. This allows the model to exploit domain knowledge that is directly applicable to all kinds of time series, from physics to finance. Called Dynamic SVMs (DSVMs), the authors' approach introduces two new control variables that need to be optimized. As a result, DSVMs have five free parameters to optimize, which means that the standard grid search (GS) method quickly becomes intractable when trying to use grids of high resolution.

Separate work has been done by the authors of [4], who propose a novel method of optimizing standard SVM parameters with a genetic algorithm (GA). Genetic algorithms provide a novel way to optimize high-degree-of-freedom models without necessarily suffering the computational complexity of grid searching. While we were unable to reproduce the results presented in their paper, we found their method compelling. Our paper combines the advancements presented in [4] and [2]; namely, we use genetic algorithms to optimize all five parameters of DSVMs.

We begin by introducing support vector regression and genetic algorithms, then we describe how our work combines the advancements in [4] and [2]. Finally, we present the data sets we used, our quantitative results and our conclusions.

II. Introduction to Support Vector Regression

This section provides an introduction to SVMs. Other excellent introductions can be found in [14], [1], [2] and [4]; this introduction is derived from them.

For a given set of data $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ with $x_i \in X \subseteq R^n$, $y_i \in Y \subseteq R$, and total number of samples l, support vector regression approximates the generating function $f(x)$ by the following form:

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

where $\phi(x)$ is a non-linear mapping of x to a high-dimensional space, brought about through a kernel function $K()$. w and b define the decision boundary hyperplane in n-space, where $w, x \in R^n$. To get optimal values for w and b, SVMs minimize what is called the "regularized risk function,"

$$minimize : \frac{1}{2}||w||^2 + C\frac{1}{l}\sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i)) \tag{2}$$

where

$$L_\varepsilon(y, f(x)) = \begin{cases} |y - f(x)| - \varepsilon, |y - f(x)| \geq \varepsilon \\ 0, otherwise \end{cases} \tag{3}$$

The *regularized risk function* is so called because it balances the *empirical risk*: $\frac{1}{l}\sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i))$, with a term that measures *generalization risk*: $\frac{1}{2}||w||^2$, e.g. a measure of how not-flat the fitted function is. Minimizing the combination of these two terms in known as the *structural risk minimization principle*, and is the primary innovation behind SVMs.

The loss function $L_\varepsilon()$ used here is the *ε-insensitive loss function* proposed by Vapnik [16] that allows for an acceptable threshold of error for each training point. This is especially useful for noisy data, to be used for example if a time series has a known instrumentation error. This loss function also allows for a sparser representation of the solution than continuous loss functions. The value C is called the regularization term, and controls the relative importance of the two individual error terms.

By introducing positive slack variables ζ_i and ζ_i^*, Equation (2) can be transformed into the primal objective function,

$$minimize: \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \tag{4}$$

subject to:
$y_i - w \cdot \phi(x_i) - b \leq \varepsilon + \zeta_i$
$w \cdot \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*$
$\zeta_i, \zeta_i^* \geq 0, \forall i$
$C > 0$

By introducing Lagrange multipliers and using the optimality constraints, we get an explicit form of Equation (1),

$$f(x) = \sum_{i=1}^{l}(a_i - a_i^*)K(x_i, x) + b \tag{5}$$

Again, $K()$ is the kernel function. In general, this can be any function that satisfies Mercer's Condition [16]. In practice, the Gaussian radial basis function (RBF),

$$K(x, y) = e^{-\gamma\|x-y\|^2} \tag{6}$$

is used most frequently for its wide applicability. The RBF kernel has one user-defined control variable, γ.

The Lagrange multipliers in Equation (5) satisfy the following conditions: $a_i \cdot a_i^* = 0$, $a_i \geq 0$ and $a_i^* \geq 0$ for $i = 1, 2, ..., l$. The values of the multipliers are found by maximizing the dual of Equation (4). The dual can be written as,

$$W(a_i, a_i^*) = \sum_{i=1}^{l} y_i(a_i - a_i^*) - \varepsilon\sum_{i=1}^{l}(a_i + a_i^*) - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(a_i - a_i^*)(a_j - a_j^*)K(x_i, x_j) \tag{7}$$

subject to:
$\sum_{i=1}^{l}(a_i - a_i^*) = 0$
$0 \leq a_i \leq C$, $0 \leq a_i^* \leq C$, for $i = 1, 2, ..., l$

The Karush-Kuhn-Tucker conditions dictate that only a few of the $(a_i - a_i^i)$ pairs will be non-zero. The training points corresponding to those values are the "support vectors:" they are the only points needed for decision making.

III. Dynamic Support Vector Regression

The first modification necessary for converting SVMs to DSVMs is to vectorize the regularization constant C shown in Equation (2) above. Equation (4) simply becomes

$$minimize: \frac{1}{2}||w||^2 + C_i\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \tag{8}$$

subject to the same constraints as above. By setting the values of C_i according to the following formula from [2]:

$$C_i = C\frac{2}{1 + exp(p_1 - 2p_1 i/l)} \tag{9}$$

and by varying the single control term p_1, the user can create a profile for C_i that is suitable for the application. Sample profiles are shown in Figure 1.

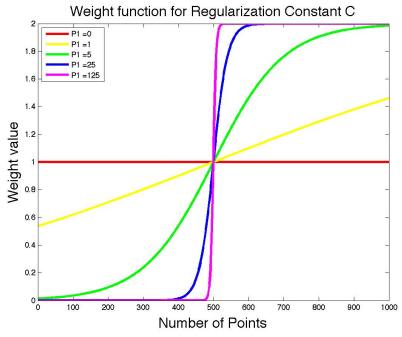

Fig. 1. Weighting for the regularization vector

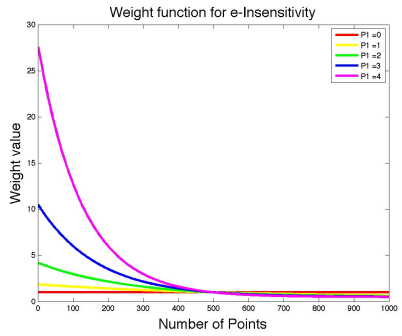

Fig. 2. Weighting for the ε-insensitivity vector

The second modification to the SVM presented in the introduction is to create a vector of values for ε, for use in the ε-insensitive loss function shown in Equation (3) above. The loss function then becomes:

$$L_\varepsilon(y_i, f(x_i)) = \begin{cases} |y_i - f(x_i)| - \varepsilon_i, |y_i - f(x_i)| \geq \varepsilon_i \\ 0, otherwise \end{cases} \tag{10}$$

The authors of [2] propose the following function for defining ε, which we adopt:

$$\varepsilon_i = \varepsilon\frac{1 + exp(p_2 - 2p_2 i/l)}{2} \tag{11}$$

Control parameter p_2 adjusts the descending profile of ε_i. The profile allows for more error in data points further back in time. Sample profiles are shown in Figure 2.

Combining these two new features, Equation (4) becomes

$$minimize: \frac{1}{2}||w||^2 + C_i\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \tag{12}$$

subject to:
$y_i - w \cdot \phi(x_i) - b \leq \varepsilon_i + \zeta_i$
$w \cdot \phi(x_i) + b - y_i \leq \varepsilon_i + \zeta_i^*$
$\zeta_i, \zeta_i^* \geq 0, \forall i$
The dual function in Equation (7) then becomes:

$$W(a_i, a_i^*) = \sum_{i=1}^{l} y_i(a_i - a_i^*) - \sum_{i=1}^{l} \varepsilon_i(a_i + a_i^*)$$
$$-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(a_i - a_i^*)(a_j - a_j^*)K(x_i, x_j) \qquad (13)$$

subject to:
$\sum_{i=1}^{l}(a_i - a_i^*) = 0$
$0 \le a_i \le C_i$, $0 \le a_i^* \le C_i$, for $i = 1, 2, ..., l$
This is precisely the formulation described in [2].

IV. Introduction to Genetic Algorithms

Genetic algorithms exploit the phenomenon of evolution for optimization. There are three major operations at work: selection, crossover and mutation. Values for each parameter of a system are stored in a vector, like genes in DNA. A large population is generated with wide variation in gene values. Each member of that population is evaluated for its "fitness" for a particular application. Fitness could be measured in running speed, weight, error rate or risk-adjusted return. Once all members of the population have been assigned a fitness, mating ensues.

A quasi-random method is used to decide which members of a population mate. The fitter an individual, the more times that individual is likely to mate, perpetuating the genes that made it fit. Generally, when two individuals mate they swap parts of their genetic material with each other, so that their offspring contain genetic material from each parent. On occasion, a mutation occurs that alters a gene to a value not directly attributable to either parent. This ensures thorough coverage of the search space. Once this new generation has been bred, each of its members is evaluated for its fitness, and the cycle repeats.

If the genetic algorithm converges to a point where all members of the population have similar genes, it implies that the population exists in a stable optimum. This optimum is not guaranteed to be global–and in general it is not. But for most practical applications a broad local optimum is better than an unstable global optimum with slightly higher fitness; a model shouldn't degrade drastically because of slight changes in the underlying system. Grid searching is not resistant to unstable solutions in the same way.

V. Genetic Algorithm Optimization of DSVMs

Our approach to prediction optimizes the DSVM's five free parameters (C, RBF γ, ε, p_1 and p_2) with a genetic algorithm like the one presented in [4]. GAs are fairly fungible in terms of their implementation; the implementation we use is outlined in pseudocode in Algorithm 1. See [6] or, preferably, our code, for detailed descriptions of each operation.

We wrote the GA library from scratch in MATLAB. We wrote the necessary modifications for DSVMs into the C/C++ LIBSVM library and associated MATLAB Mex interface by [8]. Our code and related electronic files are available upon request.

Algorithm 1 Pseudocode for Genetic Algorithm

{**Step 1: Define important variables**}
$PopulationSize = 170$
$MaxGeneration = 6$
$NumberOfGenes = 5$
$TournamentSize = 2$
$ElitistSelection = 3$
$OperateCrossover = 0.5$
$ArithmeticCrossover = 0.5$
$MutationRate = 0.1$
$GenerationNumber = 1$

{**Step 2: Generate initial population**}
Generate $PopulationSize$ individuals with gene values randomly chosen from appropriate ranges. Store this population in $CurrentPopulation$, whose size is $PopulationSize \times NumberOfGenes$

{**Step 3: Evolution**}
while $GenerationNumber \le MaxGeneration$ **do**
  Step 3.1: Calculate Fitness: For each individual in $CurrentPopulation$, calculate prediction error and save in vector $Fitness$
  Step 3.2: Tournament Selection: Select $TournamentSize$ individuals randomly from the population, and choose the fittest one. Do this $PopulationSize - ElitistSelection$ times
  Step 3.3: Elitist Selection: Choose the $ElitistSelection$-fittest individuals from $CurrentPopulation$
  Step 3.4: Crossover: Pair off the individuals chosen in Steps 3.2 and 3.3; each pair will create two children. For each pair, generate uniform random numbers a_1 and a_2 from [0,1] and do:
  if $a_1 < OperateCrossover$ **then**
    if $a_2 < ArithmeticCrossover$ **then**
      Parents create children by arithmetic crossover
    else
      Parents create children by heuristic crossover
    end if
  else
    Each parent has a child identical to itself
  end if
  Save all children to $CurrentPopulation$
  $GenerationNumber = GenerationNumber + 1$
  Step 3.5: Mutation: For each member of $CurrentPopulation$, choose a uniform random number b_1 from [0,1] and do
  if $b_1 < MutationRate$ **then**
    Mutate one of its genes
  end if
end while

VI. Experimental Setup

There are two major considerations for ensuring that a comparison of genetic algorithm (GA) to grid search (GS) optimization is fair. The first consideration springs from the fact that the GA approach does better with larger population sizes and more generations, and GS does better with more grid points. Because model construction dominates computation time, it is important that both methods be run with the same number of models. For example, if we run a grid search with 5 points on each of the 5 axes, we evaluate $5^5 = 3125$ models; to make a fair comparison to GA optimization, we ensure that the value of $PopulationSize \times GenerationSize$ is also (approximately) 3125. We have chosen to set the ratio of $PopulationSize : GenerationSize$ to roughly $25 : 1$. So, for each grid search resolution, we can calculate the number of individuals and generations to use in the comparable GA. Table I shows the values for the first few grid sizes. Note that $PopulationSize$ must be an even number in order to support two-parent mating, so the $PopulationSize$ values are rounded down to the nearest even number.

TABLE I
Grid Search and Genetic Algorithm Parameters

Grid Size:	2	3	4	...
DSVM Evaluations:	$2^5 = 32$	$3^5 = 243$	$4^5 = 1024$	...
GA PopulationSize:	16	80	170	...
GA GenerationsSize:	2	3	6	...

The second consideration for apples-to-apples comparison is that GA optimization is stochastic, whereas grid searching is generally deterministic. The GA approach can be run many times to yield a distribution of the method's performance over many trials. This is traditionally not the case for grid searching, which selects evenly-spaced points within specified ranges on each axis. A sparse grid search's performance is determined largely by chance; if one of the grid nodes happens to land in a good spot, the method will do well. If not, the whole method suffers.

In order to counter this problem and get at a meaningful comparison, we need a way to "randomize" over the distribution of viable grid searches. To do this, we randomly shift the grid points slightly along each axis, ensuring that they remain equally spaced and all within the prescribed ranges. This random shifting is done for many independent trials. Table II shows the ranges for each of the 5 parameters, which were used for both the GA and GS optimizations.

TABLE II
DSVM Parameter Ranges

Parameter:	C	RBF γ	ε	p_1	p_2
Upper Bound:	2^8	2^3	$0.05 * max$(TVD)	5	5
Lower Bound:	2^-5	2^-15	0	0	0

*TVD = Training & Validation Data

By comparing the means and standard deviations from each method over many trials, we get a clear picture of their relative reliability. This is especially meaningful when looking at performance under small numbers of model evaluations (i.e. when the grid is sparse and there are very few individuals and generations). Each method was run 20 or 100 times (based on the resultant variance) for each grid resolution (and corresponding GA) and each data set. The error means, error standard deviations and lowest individual trial errors were recorded for each test. Calculations were run on a 4-core AMD Opteron 275 machine with 4GB of RAM running Ubuntu 9.04 Linux. As noted below, one pair of tests proved too complex for this machine and needed to be broken up manually.

VII. Data Sets

We have applied the above-mentioned tests to four data sets which are commonly used in the literature. We compare the results of GA-optimized DSVMs (GA-DSVMs) and GS-optimized DSVMs (GS-DSVMs) to each other and to other published results.

The first data set was introduced by Refenes et al. [13], and is composed of seven periods of a drifting sine wave.

The second data set is the well-known annual sunspot data, which is regarded as nonlinear and non-stationary. It is widely cited, and available from the National Geophysical Data Center's website [11].

The third and fourth data sets are from the 1991-1992 Santa Fe time series prediction competition. Set "A" is a record of the intensity of a far-infrared laser in a chaotic state. It is approximately described by three coupled nonlinear ordinary differential equations. Set "D" is artificially generated, and models a nine-dimensional periodically driven dissipative dynamic system with an asymmetrical four-well potential and a drift on the parameters. A detailed description of the data sets and how they were captured can be found in [18].

VIII. Experimental Results

The GA-DSVM performed remarkably well on each of the data sets. In the case of the drifting sine wave, it was able to learn the function entirely in only a handful of model evaluations. In the case of the Sunspot and Santa Fe "A" data, the GA-DSVM found solutions with extremely low error, and converged on the answer faster than GS-DSVM. On the Santa Fe "D" data, both the GA-DSVM and GS-DSVM find excellent solutions with few model evaluations. The GA-DSVM, however, finds slightly better solutions on average, and with a much lower error variance.

A. Drifting Sinusoid Data

The drifting sine wave data originally presented by Refenes et al. [13] provide a dramatic example of the power of GA-DSVMs. Figure 3 shows the data set, whose formula is

$$y = m_1 * x + m_2 * x * sin(2\pi x) \tag{14}$$

where $m_1 = 0.1, m_2 = 0.5, 0 \leq x \leq 7$.

The data set is comprised of 700 data points. Points 1-500, 501-600 and 601-700 are the training, validation and testing sets, respectively. The previous 12 points are used to predict the next one.

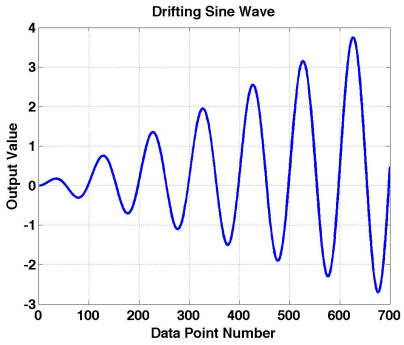

Fig. 3. Drifting Sine Wave Data Set

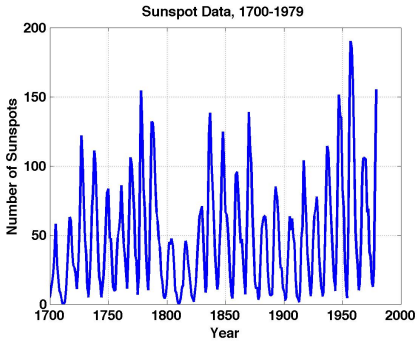

Fig. 4. Sunspot Data, 1700-1979

Both the GA-DSVM and GS-DSVM were able to learn the function, attaining negligible minimum error rates. When plotted together, the predicted and actual curves are nearly indistinguishable. On average, however, the GA-DSVM learns better solutions more consistently and has a lower minimum error. The results are shown in Table IV[1]. The GS-DSVM ran a grid search with 3 equally-spaced points per parameter, for a total of $3^5 = 243$ evaluations. The GA-DSVM ran 3 generations of 80 individuals, for a total of 240 evaluations. Because of this fast learning, a graph of the mean error convergence has been omitted.

TABLE III
DRIFTING SINE WAVE NMSE MEAN & STD.DEV. BY METHOD OVER 100 TRIALS

Method	RMSE Mean	RMSE Std. Dev.
GS-DSVMs	0.000441	0.000397
GA-DSVMs	**0.000126**	**0.000205**

TABLE IV
LOWEST DRIFTING SINE WAVE NMSE BY METHOD

Method	NMSE
Standard SVMs	0.3128 [2]
DSVMs	0.0930 [2]
GS-DSVMs	0.000005327
GA-DSVMs	**0.000002533**

B. Sunspot Data

The sunspot data are from 1700 to 1979, and are shown in Figure 4. Points 1-221, 222-256 and 257-280 are the training, validation and test sets, respectively. Again the previous 12 points are used to predict the next one. The computational efficiency of the GA-DSVM approach relative to GS-DSVMs is illuminated by this data set as illustrated in Figure 5; GA-DSVMs produce dramatically lower error rates for the same amount of computing power.

Table V shows the lowest NMSE attained by each method.

[1]The authors of [2] do not fully explain how they chose their parameters for this or any of the following data sets.

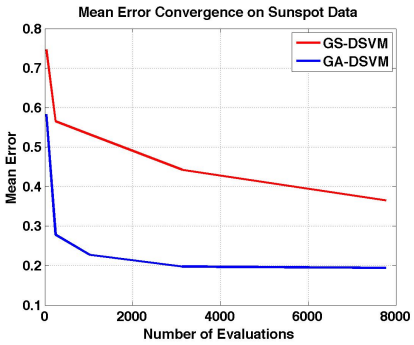

Fig. 5. Mean Sunspot Prediction Error

TABLE V
LOWEST SUNSPOT NMSE BY METHOD

Method	NMSE
Connectionist Architecture	0.35 [17]
Threshold Autoregressive	0.28 [15]
Standard SVMs	0.2682 [2]
DSVMs	0.1541 [2]
GS-DSVMs	0.1360
GA-DSVMs	**0.1080**

C. Santa Fe "A" Laser Data

The Santa Fe competition data set "A" is a record of the intensity of a far-infrared laser which is being optically pumped by another laser. The main data set contains 1000 points. A second continuation set contains the following 9093 points, for a total of 10093. Points 1-900, 901-1000 and 1001-1100 are used for the training, validation and testing sets, respectively. Eight previous data points are used to predict the next. Figure 6 shows the data.

These data also highlight GA-DSVM's computation efficiency relative to GS-DSVM. Figure 7 illustrates the drastic difference in convergence rates. The set of tests with $6^5 = 7776$ models had to be broken up into smaller parts to prevent the test machine from consuming RAM & Swap and crashing, further highlighting the value of GA-DSVM's efficient optimization.

Table VI shows the lowest NMSE attained by each method.

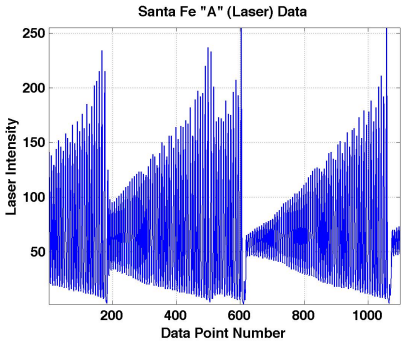

Fig. 6. Santa Fe "A" Data

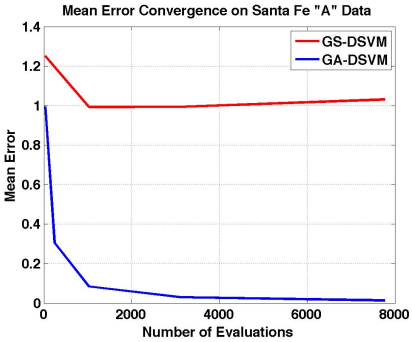

Fig. 7. Mean Santa Fe "A" Prediction Error

TABLE VI
LOWEST SANTA FE "A" NMSE BY METHOD

Method	NMSE
SLS-MDL Neural Net	0.0643 [7]
Kernel Predictive Linear Gaussian	0.026 [19]
Kernel Recursive Least Squares	0.026 [5]
Standard SVMs	0.0188 [2]
Wavelets	0.0073 [9]
GS-DSVMs	0.0069
DSVMs	0.0044 [2]
GA-DSVMs	**0.0037**

D. Santa Fe "D" Artificial Data

The Santa Fe "D" data set is artificially generated and is comprised of two 5000-point training segments plus a 500-point continuation set, for a total of 10,500 contiguous points. To be consistent with the experiments in [12], [10] and [2], points 8001-9900, 9901-10,000 and 10,000-10,025 are used for the training, validation and testing sets, respectively. These data are plotted in Figure 8. Twenty previous data points are used to predict the next.

Both the GA-DSVM & GS-DSVM quickly converged on solutions, using as few as $2^5 = 32$ evaluations each. Still, the GA-DSVM has two particular advantages. First, the standard deviation of the GA-DSVM's results is much lower than the GS-DSVM's, meaning that GA optimization is more consistent from one run to the next. Secondly, the GA-DSVM's lowest recorded error was lower than the GS-DSVM's, and in line with the lowest recorded in [2].

Table VII shows the RMSE mean and standard deviation for each method over 100 samples. Table VIII shows the lowest recorded RMSE for each, and compares them to previous benchmarks. RMSE is used for this data set in place of NMSE to make a clearer comparison to previously reported results.

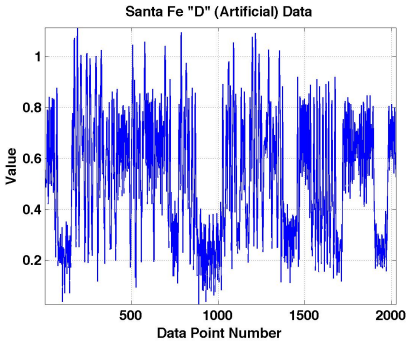

Fig. 8. Santa Fe "D" Data

TABLE VII
SANTA FE "D" RMSE MEAN & STD.DEV. BY METHOD OVER 100 TRIALS

Method	RMSE Mean	RMSE Std. Dev.
GS-DSVMs	0.0216	0.002206
GA-DSVMs	**0.0208**	**0.000778**

TABLE VIII
LOWEST SANTA FE "D" RMSE BY METHOD

Method	RMSE
Dynamics Segmentation w/ Neural Networks	0.0596 [12]
SVM Benchmark	0.0418 [10]
Dynamics Segmentation w/ SVMs	0.038 [3]
Standard SVMs	0.0212 [2]
GS-DSVMs	0.0199
GA-DSVMs	**0.0195**
DSVMs	**0.0195 [2]**

IX. CONCLUSION AND FUTURE WORK

GA-DSVMs produce well-optimized models using Mean-Square-Error-based fitness functions. They meet or receed the lowest recorded error rates for each of the four data sets explored. Furthermore, GA-DSVMs are significantly more computationally efficient than GS-DSVMs; for any given number of model evaluations, it is better to use GA optimization than GS optimization. Finally, GA optimization is more reliable from one trial to the next, based on its lower standard deviation of error rate. In summary, GA-DSVMs do better than GS-DSVMs under all situations we explored.

Our results show that the choice of an optimization method can be almost as important as the choice of the learning method. Future work would explore this relationship further.

REFERENCES

[1] L.J. Cao, F.E.H. Tay, *Financial Forecasting Using Support Vector Machines*, Neural Computing & Applications, Volume 10, Number 2, May 2001, Pages 184-192.

[2] L.J. Cao, Q. Gu, *Dynamic support vector machines for non-stationary time series forecasting*, Intelligent Data Analysis, Volume 6, Number 1, 2002, Pages 67-83.

[3] M. W. Chang, C. J. Lin and R. C. Weng, *Analysis of switching dynamics with competing support vector machines*, IEEE Transactions on Neural Networks, Volume 15, Issue 3, May 2004, Pages 720-727.

[4] Y. Dong, M. Tu, Z. Xia, G. Xing, *An Optimization Method for Selecting Parameters in Support Vector Machines*, Sixth International Conference on Machine Learning and Applications, December 13-15, 2007, Pages 1-6.

[5] Y. Engel, S. Mannor and R. Meir, *The kernel recursive least squares algorithm*, IEEE Transactions on Signal Processing, Volume 52, 2004, Pages 2275-2285.

[6] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, 2nd ed. New York, NY, Wiley-Interscience, 2004

[7] Y. N. Lai and S. Y. Yuen, *Successive-Least-Squares Error Algorithm on Minimum Description Length Neural Networks for Time Series Prediction*, Proceedings of the 17th International Conference on Pattern Recognition, Volume 4, 2004, Pages 609-612.

[8] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM : a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[9] R.L. Milidiu, R.J. Machado and R.P. Rentera, *Time-series forecasting through wavelets transformation and a mixture of expert models*, Neurocomputing, Volume 28, 1999, Pages 145-156.

[10] K.R. Muller, A.J. Smola, G. Ratsch, B. Scholkopf and J. Kohlmorgen, *Using support vector machines for time series prediction*, Advances in Kernel Methods Support Vector Learning, B. Scholkopf, C.J.C. Burges and A.J. Smola, eds, 1999, Pages 243-254.

[11] http://www.ngdc.noaa.gov/stp/SOLAR/ftpsunspotnumber.html Accessed June 18th 2009.

[12] K. Pawelzik, K.R. Muller and J. Kohlmorgen, *Annealed competition of experts for a segmentation and classication of switching dynamics*, Neural Computation, Volume 8, 1996, Pages 340-356.

[13] A.N. Refenes, Y. Bentz, D.W. Bunn, A.N. Burgess and A.D. Zapranis, *Financial time series modeling with discounted least squares backpropagation*, Neurocomputing, Volume 14, 1997, Pages 123-138.

[14] F.E.H. Tay, L.J. Cao, *Application of support vector machines in financial time series forecasting*, Omega: The International Journal of Management Science, Volume 29, Issue 4, August 2001, Pages 309-317.

[15] H. Tong and K.S. Lim, *Threshold autoregressive, limit cycles and cyclical data*, Journal of Royal Statistical Society, B42(3), 1980, Pages 245-292.

[16] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY: Springer, 1999.

[17] A.S. Weigend, B.A. Huberman and D.E. Rumelhart, *Predicting the future: a connectionist approach*, International Journal of Neural System, Volume 1, 1990, Pages 193-209.

[18] A. S. Weigend and N. A. Gershenfeld, eds. *Time Series Prediction: Forecasting the Future and Understanding the Past*, Reading, MA: Addison-Wesley, 1994.

[19] D. Wingate and S. Singh, *Kernel Predictive Linear Gaussian models for nonlinear stochastic dynamical systems*, Proceedings of the 23rd international Conference on Machine Learning, Volume 148, 2006, Pages 1017-1024.