Comparison of Data Reduction Techniques Based on SVM Classifier and SVR Performance

Dave Zhao, Ramona Georgescu, Peter Willett

Elec. and Comp. Engineering Department, University of Connecticut, Storrs, CT 06269 {da-wei, ramona, willett}@engr.uconn.edu

August 18, 2011

ABSTRACT

In this work, we applied several data compression techniques to simulated data and the Turbofan engine degradation simulation data set from NASA, with the goal of comparing their performance when coupled with the Support Vector Machine (SVM) classifier and the SVM regression (SVR) predictor. We consistently attained correct rates in the neighborhood of 90% for simulated data set, with the Principal Component Analysis (PCA), Sparse Reconstruction by Separable Approximation (SpaRSA) and Partial Least Squares (PLS) having a slight edge over the other data reduction methods for data classification. We achieved 22% error rate with SRM for the Turbofan data set 1 and 40% error rate with PCA for Turbofan data set 2. Throughout the tests we have performed, PCA proved to be the best data reduction method.

Keywords: Data reduction, PCA, PLS, SRM, OMP, SpaRSA, Classification, Regression, SVM, SVR

1. INTRODUCTION

As technology advances, data collection capability has been improved in terms of both accuracy and quantity. However, the bandwidth of the information transmission is still limited. Therefore, as we are capable to collect more and more information, data reduction before transmission has become even more crucial to reduce the transmission cost. A slight loss in data processing performance in exchange for much lower data transmitting cost is often tolerated in many applications. Most of the time, the information obtained is not always 100% useful, i.e. not all the features of the data are relevant [16]. For the same reason, in classification, it is desirable to reduce the dimensionality of the data and reconstruct them from a lower dimensional samples. In SVR, it is also noticeable that we can achieve very good performance with much lower dimensional data sample.

In our simulated Gaussian mixture (GM) data set, we generated 4 classes, each consisting of a Gaussian mixture (GM) of 2 elements. 400 measurements were constructed for each data set with each measurement consisting of 30 features. We applied the dimensionality reduction techniques to three sets of data, with increasing difficulty of classification. We employed Principal Component Analysis (PCA), Partial Least Squares (PLS), Structurally Random Matrices (SRM), Orthogonal Matching Pursuit (OMP) and Sparse Reconstruction by Separable Approximation (SpaRSA) to compress the data set and applied the Support Vector Machines (SVM) and Proximal Support Vector Machine (PSVM) classifiers to the results of the compression to determine the class categories.

We also constructed a Support Vector Regression data set to test with the same data compression techniques and classifiers. The purpose of the SVR data set is to represent the time to failure of a particular system. Each class of the data set represents a health stage of the system. For testing purposes, we started with 5 classes. Each class consists of Gaussian random numbers with random mean and unit variance. We used the suboptimal nearest neighbor solution to the traveling salesman problem (TSP) solution to rearrange the classes starting with a randomly picked class. Data points for the SVR data set were generated along the path given by the TSP solution previously. We then tested the data set with our data reduction techniques coupled with SVM classifier and SVR predictor. As expected, SVM had a difficult time to separate the classes. For SVR prediction, we were interested in the specific system health value, so instead of determining which stage the system is in, we are trying to predict the remaining useful life (RUL) of the system at any given point.

Finally, we challenged our techniques with the difficult Turbofan engine degradation simulation data set and applied SVM classification and SVR prediction.

2. DATA SETS

2.1 GM Data Generation

The dimensionality reduction techniques and the SVM classifier have been implemented on three different GM data sets of varying levels of difficulty. Each data set had 4 classes with 100 measurements per class. Each measurement had 30 features. Each class was constructed from a Gaussian mixture of two Gaussians with distinct mean and variance. Two thirds of the measurements went into the training set and one third was set aside for testing. The design parameter of the data sets are shown in Table 1.

In the first data set, we separate all four classes far apart so that we can easily determine the class category. The means and variances of the Gaussian mixture for each class were fixed. The variances for the Gaussian elements were the same for all classes and relatively small. In the second data set shown in Figure 1, the means and variances of the Gaussian mixtures were randomly generated by using Gaussian random number generator (GR). The variances were still the same across classes and relatively small compared to the means. Finally, in the third data set shown in Figure 2, we used different means and different variances for all the Gaussian mixtures, also the variances were much larger than before.

Table 1: GM data sets design parameters

Parameters	Mean	Variance	Classes	GMs	Features	Data points	LRT
Data set 1	Fixed	25	4	2	30	400	100%
Data set 2	GR from 0 to 100	25	4	2	30	400	100%
Data set 3	GR from 0 to 100	1600	4	2	30	400	99.34%

2.2 SVR Data Generation

We constructed our SVR data set in two steps. First, a Travelling salesman problem (TSP) was formulated with our multiple classes. The idea is to find the ”shortest path” from any given class to the rest of classes. We then applied smoothing techniques to smooth out the means by using a mixing coefficient. For simplicity we started with 5 classes shwon in Figure 3, each class consisting of Gaussian variables with random means and unit variance. The distance between any two classes

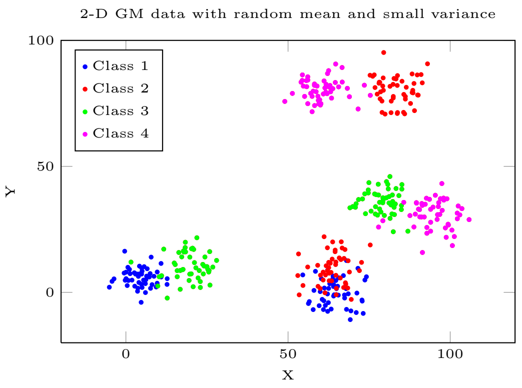

Figure 1: GM data set in a 2 dimensional space. We implemented 4 classes, each has 2 random means and relative small covariance.

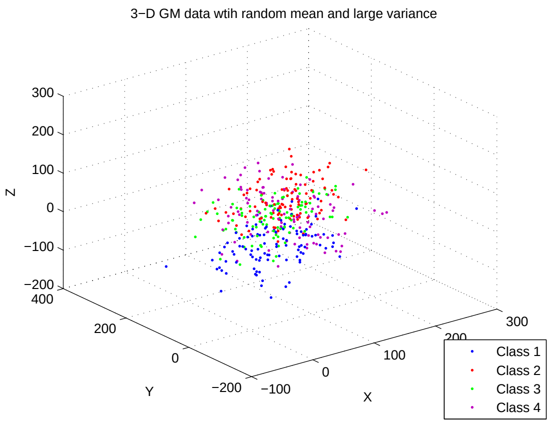

Figure 2: GM data set in a 3 dimensional space. For this set up, we implemented 4 classes, each has 2 random means and relatively large covariance.

has been calculated by using the Integral Square Difference (ISD)[17] , which is a cost function able to be evaluated in closed form. Expanding the ISD distance measure equation yields the following terms:

$$\begin{aligned} J_S = \int & [f\{X(k)|\Omega_{N_h}(k)\}^2 \\ & - 2f\{X(k)|\Omega_{N_h}(k)\}f\{X(k)|\bar{\Omega}_{N_r}(k)\} \\ & + f\{X(k)|\bar{\Omega}_{N_r}(k)\}^2]dX(k) \end{aligned} \tag{1}$$

where $N_h(k)$ represents joint events used in [17], $\Omega_{N_k}(k)$ represents the parameters of the $N_h(k)$. We then rewrite Equation 1 as

$$J_S = J_{hh} - 2J_{hr} + Jrr \tag{2}$$

where J_{hh}, J_{hr}, and J_{rr} are defined as

$$J_{hr} = \int f\{X(k)|\Omega_{N_h}(k)\}f\{X(k)|\bar{\Omega}_{N_r}(k)\}dX(k) \tag{3a}$$

$$J_{rr} = \int f\{X(k)|\bar{\Omega}_{N_r}(k)\}^2 dX(k) \tag{3b}$$

$$J_{hh} = \int f\{X(k)|\Omega_{N_h}(k)\}^2 dX(k) \tag{3c}$$

Since we are using Gaussian mixtures, the equation can be rewritten as:

$$f\{X(k)|\Omega_{N_h}(k)\} = \sum_{i=1}^{N_h(k)} p_i \mathcal{N}\{X; \mu_i, P_i\} \tag{4a}$$

$$f\{X(k)|\bar{\Omega}_{N_r}(k)\} = \sum_{i=1}^{N_r(k)} \bar{p}_i \mathcal{N}\{X; \bar{\mu}_i, \bar{P}_i\} \tag{4b}$$

where $\{p_i, \mu_i, P_i\}$ are the weights, means and covariances of the Gaussian functions in a mixture, and $\{\bar{p}_i, \bar{\mu}_i, \bar{P}_i\}$ are the same parameters for the second Gaussian mixture. Substituting these expressions into the block of Equations 3 and reversing the order of integration and summation:

$$J_{hr} = \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i \bar{p}_j \int \mathcal{N}\{X; \mu_i, P_i\} \mathcal{N}\{X; \bar{\mu}_j, \bar{P}_j\} dX(k) \tag{5a}$$

$$J_{rr} = \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} \bar{p}_i \bar{p}_j \int \mathcal{N}\{X; \bar{\mu}_i, \bar{P}_i\} \mathcal{N}\{X; \bar{\mu}_j, \bar{P}_j\} dX(k) \tag{5b}$$

$$J_{hh} = \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i p_j \int \mathcal{N}\{X; \mu_i, P_i\} \mathcal{N}\{X; \mu_j, P_j\} dX(k) \tag{5c}$$

As derived in [18], the product of two gaussian PDFs, which forms the basic building block of Equations 5, can be simplified to the following form:

$$\mathcal{N}\{x; \mu_1, P_1\} \mathcal{N}\{x; \mu_2, P_2\} = \alpha \mathcal{N}\{x; \mu_3, P_3\} \tag{6}$$

where

$$\alpha = \mathcal{N}\{\mu_1; \mu_2, P_1 + P_2\} \tag{7a}$$

$$P_3 = (P_1^{-1} + P_2^{-1})^{-1} \tag{7b}$$

$$\mu_3 = P_3(P_1^{-1}\mu_1 + P_2^{-1}\mu_2) \tag{7c}$$

We then substitute Equation 7 into Equation 3 and get

$$J_{hr} = \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i \bar{p}_j \mathcal{N}\{\mu_i; \bar{\mu}_j, P_i + \bar{P}_j\} \tag{8a}$$

$$J_{rr} = \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} \bar{p}_i \bar{p}_j \mathcal{N}\{\bar{\mu}_i; \bar{\mu}_j, \bar{P}_i + \bar{P}_j\} \tag{8b}$$

$$J_{hh} = \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i p_j \mathcal{N}\{\mu_i; \mu_j, P_i + P_j\} \tag{8c}$$

We compute the "shortest distance" also known as correlation between any two classes using

$$corr = \frac{J_{hr}}{\sqrt{J_{hh} J_{rr}}} \tag{9}$$

To solve the Travelling salesman problem (TSP), we used the nearest neighbor method. We first randomly picked a class (class A), then we computed all the distances between class A and all the other classes. We selected the next class (class B) with the shortest distance from class A. Once we picked class B, we computed the correlation between class B and the rest of classes (not including class A). This process repeats until we have selected all classes.

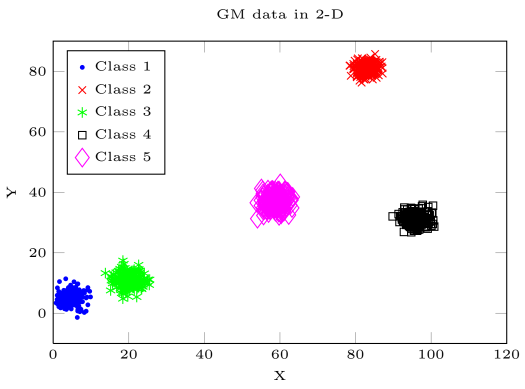

Figure 3: The SVR data set consists of 5 classes, each class has a random mean. We kept the variance relative small for demonstration purpose

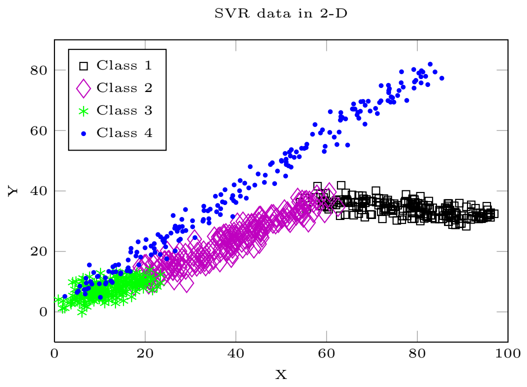

Figure 4: The 2-D view of the SVR data set. We started with a randomly picked class, in this example it is class 4 in Figure 3. We then generated smoothed the data points to the nearest class. Similarly, we generated instances from the rest of the classes.

Since we adopted multi-dimensional Gaussian mixtures for our data set, it is desirable to reorder the elements with the multidimensional means, across all the mixtures in terms of distances between them. We used the Auction algorithm [24], a combinatorial optimization algorithm to compute the closest pairs of elements in term of minimizing their assignment costs in a 2-D space. For example, if 3-dimensional Gaussian are used, a possible Auction assignment could be: μ_{11} to μ_{21}, μ_{12} to μ_{23} and μ_{13} to μ_{22}, where μ_{ij} stands for the i^{th} means's in j^{th} element.

A mixing coefficient was then computed using

$$\lambda = \frac{y - y_0}{y_1 - y_0} \tag{10}$$

where y is the uniform randomly generated time to failure in the interval from y_0 to y_1 and y_0 and y_1 are the lower and upper bounds of its class boundary. Finally, we generate GM data points based on the new means $\lambda\mu_1$ and $(1-\lambda)\mu_2$ and fixed variance shown in Figure 4.

2.3 Turbofan degradation simulation data set

The Turbofan degradation simulation data set is a run to failure data set, which was generated by using the C-MAPSS tools[19]. Four different sets were simulated under different combinations of operational conditions and faults modes. 21 sensors channels along with 3 input parameters were recorded to characterize fault evolution. Each data set consists of training, testing and remaining useful life subsets. Each set also includes multiple engines RUL of the same type. The engines operate normally at the start and develop a fault at some point during the measurements. In the training set, the fault grows in magnitude until system failure. In the test set, the measurement stops some time prior to system failure. The objective is to predict the remaining useful life of each engine.

Based on the remaining useful life of the system in the training set (Figure 5), each measurement was assigned to a class by comparing its RUL to a threshold. The x-axis is the raining useful life

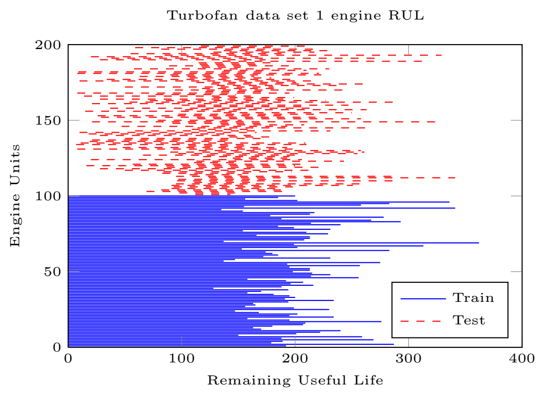

Figure 5: Remaining useful life of each engine in Turbofan data set 1 for both training set and testing set. It was used to determine the class range assignment for the SVM classifier. The engine units from 100 to 200 on the figure, are actually the RUL for engine units 1 to 100 in the testing set for clarity.

of the unit, the y-axis represents the unit number. There are 100 units, the blue ones are from the training set and the red ones are from the testing set. We tested with 2 classes and 4 classes. We applied data compression techniques to reduce the dimension of the data. Here, we selected 3 inputs along with 21 sensor measurements as our data components. SVM was applied to determine the class category of the instances in the test set.

3. DATA REDUCTION TECHNIQUES

3.1 Principal Component Analysis

Principal Components Analysis (PCA) is a useful statistical technique that has many application in data with high dimensions, such as facial recognition, image compression, pattern learning, etc. The patterns of a data set can be hard to find in high dimensions. PCA expresses the data in a way such to highlight similarities and differences; it identifies the patterns in the data [22]. Once we determined the data pattern, we can reduce the data dimension, i.e. compress the data as much as possible without losing key information.

Let X be a n-by-p data matrix, where n is the numbers of measurements and p is the numbers of features of the data set. PCA is carried out with 4 steps. It first computes the mean of each dimension and subtracts it from each of the data dimensions. It then computes the p-by-p covariance matrix. In the next step, PCA calculates the unit eigenvectors and eigenvalues of the covariance matrix. Taking the eigenvectors of the covariance matrix helps PCA extracts the characteristics of the data. The principle component of the data set is determined by the eigenvector with the highest eigenvalue, which represents the most significant relationship between the data dimensions. Next, PCA re-ranks the data set dimensions based on the order of their corresponding eigenvalues form the highest to lowest. PCA also groups their corresponding eigenvectors in the feature vector, also known as the loadings of the data set. Finally, PCA computes the scores of the data set by multiplying the original data with the transposed feature vector, i.e. transforming the original data set into the space of the principal components. So, we have

$$X_S = XX_L^T \tag{11}$$

where X_S is the score of the data set and X_L is the loadings. One property of PCA's loadings is that its inverse is also its transpose, because the loadings are given by unit eigenvectors [22]. Therefore, we get

$$X = X_S X_L^T \tag{12}$$

In our implementation, we used the *princomp* function in the Statistics Toolbox of MATLAB® (R2009b) to implement PCA and calculate the scores and loadings of the training set. We project the measurements in the testing set on the loadings obtained from the training set to obtain the scores for the testing set. We then pass the scores from the testing set to the chosen classifier and/or predictor.

3.2 Partial Least Squares

Partial Least Square (PLS) is an extension of the multiple linear regression model, it assumes linear model describes the linear relationship between a depended (response) variable and a set of independent (prediction) variables. Often during data prediction of a large data set, we are likely to obtain a model that fits the sampled data very well, however fails to predict the information we are interested in due to over-fitting. Even though the data set could be very large in terms of features, there may be only a few dimensions that account for most of the variation in the response. PLS will extract those, while making sure to model the responses well.

For a $(n \times p)$ data matrix X, in which each row represents an observation and each column represents a feature, and a $(n \times 1)$ vector Y that contains the corresponding classes for the observations in X, the PLS transformation is given by:

$$X = X_S X_L^T + error_X \tag{13}$$

$$Y = Y_S Y_L^T + error_Y \tag{14}$$

With n_{Comp} being the number of components retained, X_S is a matrix of dimension $(n \times n_{Comp})$ representing the extracted predictor scores and X_L is a matrix of dimension $(p \times n_{Comp})$ containing the predictor loadings. Y_S is of dimension $(n \times n_{Comp})$ and Y_L is of dimension $(1 \times n_{Comp})$.

PLS uses the nonlinear iterative partial least squares (NIPALS) algorithm [4] to compute the score matrix $T = XW$ for an appropriate weight matrix W and c such that $[cov(x_s, y_s)]^2 = [cov(Xw, Yc)]^2$ where $x_s = Xw$, $y_s = Yc$ and $cov(x_s, y_s) = x_s^T y_s / n$ denotes the sample covariance between the score vectors x_s and y_s. In other words, PLS extracts components from X that are highly correlated with Y by performing a simultaneous decomposition for X and Y while trying to explain the covariance between them as much as possible. PCA uses the weight to reflect the covariance structure between the predictor variables, while PLS uses it to reflect the covariance structure between predictor and response variables[23].

Several variants of PLS exist; we used the SIMPLS form of PLS introduced by de Jong [6] in which X_S is an orthonormal matrix while Y_S is neither orthogonal nor normalized.

The matrix constructed by the weight vectors w is a $(p \times n_{Comp})$ matrix W of PLS weights with the property that $X_S = XW$.

We used the *plsregress* function in the Statistics Toolbox of MATLAB® (R2009b) to implement PCA and calculate the scores and loadings of X and Y in the training set. We pass the scores obtained on the training set to the chosen processor along with the scores obtained for the testing set by projecting the observations in the testing set (X_{test}) on the loadings of the training set (X_L) according to the formula:

$$X_S = X_{test}W(X_L^TW)^{-1}. \tag{15}$$

.

3.3 Structurally Random Matrices

Compressive sensing is a recent theory [7, 8, 9] that predicts that sparse vectors in high dimensions can be recovered from incomplete information. The main motivation is that many real-world signals can be well approximated by an expansion in terms of a suitable basis, which has only a few non-zero terms, a sparse basis that is. The idea is to obtain a compressed format which computes the coefficients in the basis and only keeps the largest coefficients.

When an input signal is of large length, such as JPEG or MP3, using a random projection is clearly impractical as a large amount of computational complexity and memory buffering is needed to compute the projection. Recently, structurally random matices (SRMs) have been proposed as a fast and highly efficient compressed sensing method that guarantees optimal performance[16].

We use the definition of a structurally random matrix Φ as a product of three matrices:

$$\Phi = \sqrt{\frac{d}{M}}DFR \tag{16}$$

where R, the local randomizer, is a $d \times d$ random diagonal matrix whose diagonal entries R_{ii} are i.i.d. Bernoulli random variables with $P(R_{ii} = \pm 1) = \frac{1}{2}$, F is a $d \times d$ orthonormal matrix whose absolute magnitude of all entries are on the order of $O\left(\frac{1}{\sqrt{d}}\right)$ (in practice, F is chosen to have fast computation and efficient implementation such as fast Fourier transform (FFT), discrete cosine transform (DCT), etc.) and D, a uniformly random downsampler, is a matrix composed of nonzero rows of a random diagonal matrix whose diagonal entries D_{ii} are i.i.d. binary random variables with $P(D_{ii} = 1) = \frac{M}{d}$ (D contains M nonzero rows and thus, Φ is a $M \times d$ matrix). Thus, SRM is also a promising dimensionality reduction transform which can be shown to preserve all pairwise distances of high dimensional vectors to within an arbitrarily small factor.

We used the same R on the training set and the testing set, which was fixed a priori. We chose the discrete cosine transform for F as it gives real outputs. We also used randomly generated coefficients for each instance and kept the same order coefficients for the training set and the testing set [10].

3.4 Orthogonal Matching Pursuit

Matching pursuit (MP) finds the "best matching" projections of a multi-dimensional data onto a over complete redundant set of data (dictionary). It solves for x the underdefined equation $y = Ax$.

MP is an iterative greedy algorithm that selects at each step the column which is most correlated with the current residuals. More specifically, at the p^{th} iteration, the selected column (a_{s_p}) of A is given by:

$$s_p = \arg\max_j \frac{|a_j^h r_{p-1}|^2}{||a_j||^2} \tag{17}$$

for j not in the index set of all previously selected columns $I_{p-1} = \{s_1, ...s_{p-1}\}$. Then, $\hat{x}_{s_p}$, the element of $\hat{x}$ associated with a_{s_p}, is found as follows:

$$\hat{x}_{s_p} = \frac{a_{s_p}^h r_{p-1}}{||a_{s_p}||^2}. \tag{18}$$

The updated residual vector is computed as:

$$r_p = r_{p-1} - \hat{x}_{s_p} a_{s_p} \tag{19}$$

where we initialize $r_0 = y$.

However, there is one issue with MP. The set of coefficients $\hat{x}_{s_p}$ may not give the minimal residual error when the set of columns are not orthogonal. This is because r_p is not necessarily orthogonal to a_{sj} for $j < p$ even if it is orthogonal to a_{s_p}. It means that $\hat{x}_{s_p}$ from Eq. 18, the residual vector r_p is not orthogonal to the space spanned by a_{s_j} $j \leq p$ and it is not of minimal length.

The Orthogonal Matching Pursuit (OMP) algorithm was created to address this issue. At each step after a new column has been chosen according to Eq. 17, the OMP algorithm recomputes the coefficients as follows:

$$\hat{x}_{s_p}^{\text{OMP}} = \arg\min_x ||y - A_{s,p}x||^2 \tag{20}$$

$$= [A_{s,p}^h A_{s,p}]^{-1} A_{s,p}^h y \tag{21}$$

where $A_{s,p} = [a_{s_1} ... a_{s_p}]$ and ^{h} is the Hermitian operator (i.e. the conjugate transpose of a matrix). This equation essentially carries out the least squares minimization based on the set of columns chosen at each iteration. However, the rule of selecting the new column is the same as for the MP [12].

In our case, we first create A as a matrix populated by i.i.d. normal distributed random numbers. We then run OMP for each observation in the training set and in the testing set. For each observation we obtain a longer vector that contains a small number of nonzero entries. The number of nonzero entries is specified such that dimensionality reduction is achieved.

Only the nonzero elements and their location are transmitted, therefore we are doubling the communication cost, but achieving dimensionality reduction nonetheless.

3.5 Sparse Reconstruction by Separable Approximation

A common problem in signal processing is to find the sparse approximated solution for a large linear system of equations. One standard approach is to minimize an objective function that includes a quadratic error term added to a sparsity-inducing regularizer [20]. An algorithmic frame work was created to minimize the sum of a smooth convex function and a nonsmooth nonconvex regularizer.

During each iteration, an optimization subproblem involving a quadratic term with diagonal Hessian plus the original sparsity-inducing regularizer was solved. Sparse reconstruction by separable approximation (SpaRSA) proposes an approach for solving unconstrained optimization problems of the form

$$\min_x \phi(x) := f(x) + \tau c(x); \tag{22}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function, and $c : \mathbb{R}^n \to \mathbb{R}$, usually called the regularizer or regularization function, is finite for all $x \in \mathbb{R}^n$, but usually nonsmooth and possibly also nonconvex. Equation 22 generalizes the minimization of

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \tau \|\mathbf{x}\|_1 \tag{23}$$

where $y \in \mathbb{R}^k$, $A \in \mathbb{R}^{k \times n}$ (usually $k < n$), $\tau \in \mathbb{R}^+$, $\|y - Ax\|_2$ is the l_2 norm, which defined as $\|x\|_2 = (\sum_i |x_i|^2)^{1/2}$. The approach to solving problems of the form Equation 22 is tailored to problems in which the following subproblem can be set up and solved efficiently at each iteration:

$$\mathbf{x}^{t+1} arg \min_{\mathbf{z}} (\mathbf{z} - \mathbf{x}^t)^T \nabla f(\mathbf{x}^t) + \frac{\alpha_t}{2} \|\mathbf{z} - \mathbf{x}^t\|_2^2 + \tau c(\mathbf{z}) \tag{24}$$

for some $\alpha_t \in \mathbb{R}^+$. It is much less expensive to compute the gradient ∇f and to solve Equation 24 than it is to solve the original problem in Equation 22. An equivalent form of subproblem for Equation 23 is

$$\mathbf{x}^{t+1} \in arg \min_{\mathbf{z}} \frac{1}{2} \|\mathbf{z} - \mathbf{u}^t\|_2^2 + \frac{\tau}{\alpha_t} c(\mathbf{z}) \tag{25}$$

where

$$\mathbf{u}^t = \mathbf{x}^t - \frac{1}{\alpha_t} \nabla f(\mathbf{x}^t) \tag{26}$$

This form is named iterative shrinkage/thresholding (IST) algorithm in the literature. The algorithm also looks for a c which is separable into the sum of functions of the individual components of its argument, that is,

$$c(\mathbf{x}) = \sum_{i=1}^{n} c_i(x_i) \tag{27}$$

The l_1 regularizer in Equation 23 obviously has this from (with $c_i(z) = |z|$).

In general, the approach requires solution of Equation 24 at each iteration. When the regularizer c is separable or group-separable, the solution of Equation 24 can be obtained from a number of scalar minimizations, whose solution are often available in closed form.

In our study, we used the package from [20] for MATLAB® (R2009b). We feed in normalized data, the same dictionary matrix (Gaussian random numbers) used in OMP, and a compression parameter τ, which influences the sparsity of the returning data. We then transform them onto the space described by the dictionary. Finally, we feed the resulting data into the chosen processor to complete data prediction.

4. CLASSIFIER AND PREDICTOR

4.1 Support Vector Machines

A Support Vector Machine (SVM) is a classifier designed to separate data into two classes. It relies on preprocessing the data to represent patterns in a high dimension space, usually much higher than the original feature space. With an appropriate nonlinear mapping to a sufficient high dimension space, data can be separated with a hyperplane [25]. SVM can also apply nonlinear classification using kernels.

The equation of a general hyperplane can be described as $w'x + b = 0$ with x being a point (a vector) and w weights (also a vector). In the case of linear classification, after mapping and applying the hyperplane, one class is determined by all the x_k which give $w'x_k + b > 0$, and the other class is determined by all the x_j which give $w'x_j + b < 0$. Most of the time, the dimensionality of the mapped space can be arbitrarily high, and it is preferable to the SVM to select the dimensionality that makes the distances from the hyperplane to the closest data point be as large as possible.

In practice, it is unlikely that a line can perfectly separate the data. When there is a curvy decision boundary, adopting a "line" solution is probably not desirable as the data could have large noise and outliers. So, a smooth decision boundary that ignores a few data points is better than one that tries to classify all the points and loops around the outliers. Adding slack variables allows a point to be a small distance on the wrong side of the hyperplane without violating the problem constraints. An optimization-based derivation of SVMs is presented in the tutorial in [13].

In our simulation, we needed to perform multiclass SVM classification and thus, we applied the technique of one vs. the rest. We have 4 classes and therefore we need to run 4 SVM classifiers. The first classifier tries to distinguish between instance of class 1 on one hand and instances of classes 2, 3 and 4 on the other hand. The second classifier focuses on class 2 vs. the other classes and so on. With this method, for each test instance, we obtain 4 classifications from the 4 SVM classifiers. Each SVM leads to a decision function of the form [14]:

$$f(x) = sgn\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right) \tag{28}$$

where m is the number of input instances x, y are the class labels, α are the Lagrange multipliers and k is the chosen kernel. The final classification decision is taken according to the maximal output before applying the sign function [14]:

$$\arg \max_{j=1,\ldots,M} g^j(x), \tag{29}$$

where

$$g^j(x) = \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \tag{30}$$

4.2 Proximal Support Vector Machine

The SVM classifier assigns instances to one of two disjoint halfspaces. In PSVM, the separating planes are not bounding planes, but can be thought of as proximal planes, around which the instances of each class are clustered, and which are pushed as far apart as possible [16]. This formulation can also be interpreted as regularized least squares.

PSVM leads to a fast and simple algorithm for generating a linear or nonlinear classifier that only requires the solution of a single system of linear equations. In contrast, SVM solves a quadratic or linear program that requires considerable longer computational time. Computational results on publicly available data sets indicate that the proximal SVM classifier has comparable test set correctness to that of the SVM classifier [30].

4.3 Support Vector Machine Regression

SVMs can also be applied to regression problems by introducing an alternative loss function [26]. Generally speaking, there are three types of loss function: least squares error (LSE), Laplacian and Huber. Each of them has advantages, e.g. LSE is sensitive to outliers, while the Laplacian is not, and Huber is a robust loss function that works well when the underlying distribution of the data is unknown [26]. However, all these loss functions produce no sparseness in the support vectors. Smola [26] proposed another loss function which address this issue, the ϵ-insensitive loss function.

In ϵ-support vectors regression (SVR), the goal is to find a function that has at most ϵ deviation away from the actual training data, and also it needs is as flat as possible, i.e., any error less than ϵ is acceptable [28].

Consider a linear function

$$f(x) =< w, x > +b \tag{31}$$

with $w \in X, b \in \mathbb{R}$, where $< ., . >$ denotes the dot product in X. Flatness of this equation means that we prefer a small w [28]. So, we can rewrite this equation as a optimization problem to minimize $\|w\|^2$:

$$\begin{aligned} &\min \quad \frac{1}{2}\|w\|^2 \\ &s.t. \quad \begin{cases} y_i - < w, x_i > -b & \leq \epsilon \\ < w, x_i > +b - y_i & \leq \epsilon \end{cases} \end{aligned} \tag{32}$$

Equation 32 indicates that all pairs (x_i, y_i) are within ϵ precision for function f [28]. However, one needs to leave some room for errors, namely adding some slack variables (ξ_i and ξ_i^*). So, Equation 32 can be rewritten as

$$\begin{aligned} &\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(x_i + x_i^*) \\ &s.t. \quad \begin{cases} y_i - < w, x_i > -b & \leq \epsilon + \xi_i \\ < w, x_i > +b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \end{aligned} \tag{33}$$

The constant $C > 0$ determines the tradeoff between the flatness of f and the amount of additional error larger than ϵ which is tolerated. The ϵ-loss function corresponding to in Equation 33 has the form:

$$|\xi|_\epsilon := \begin{cases} 0 & \text{if} |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{otherwise} \end{cases} \tag{34}$$

The optimization problem in Equation 33 can be solved easily in its dual formulation [28].

We used the LIBSVM package in our study [29], written in C code which also runs in MATLAB® (R2009b). Similar to SVM, it has two major components: SVR-train and SVR-predict. SVR-train takes in training data which were preprocessed with all the data reduction techniques and different compression rates. SVR-predict takes in the results from SVR-train and the preprocessed testing data. SVR returns predicted values for each measurement.

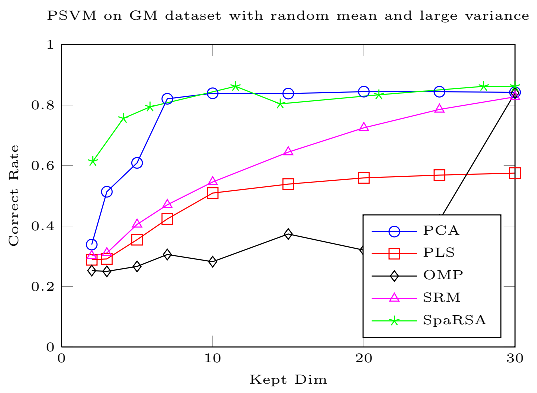

Figure 6: Performance of PSVM classifier for Gaussian Mixture data set with random means and large variance. The result indicates that we can achieve at most 80% correct rate while LRT results a correct rate of 99% for this specific data set. PCA and SpaRSA are very good methods for this set up, both of them gave good classification results at high compression.

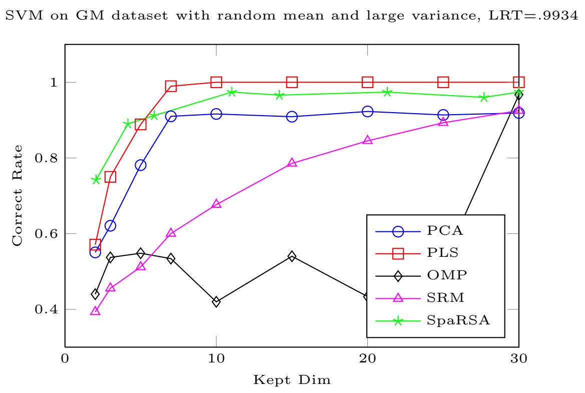

Figure 7: Performance of SVM classifier for Gaussian Mixture data set with random mean and large variance. PLS achieved the best performance when 10 dimension are kept. PLS, SpaRSA and PCA performed better coupled with a SVM classifier than PSVM for all different compression rates with mostly a correct rate of 90% or better.

5. RESULTS

5.1 PSVM vs. SVM on GM data set

50 Monte carlo runs were performed for both SVM and PSVM under all 3 scenarios, we also performed likelihood ratio tests (LRT) for each scenario to determine the optimal classification performance. For all test cases, SVM performs better than PSVM for all the data compression techniques. For the compression technique with the worst performance (OMP), there is a large difference between using SVM (Figure 7) and PSVM (Figure 6). The correct rate was improved by about 20% on average for all the data reduction techniques when we applied SVM instead of PSVM. For those compression techniques with good performances, there is not a large difference between the two classifiers. PSVM is helpful in improving processing time efficiency. SVM gives better results on all scenarios.

PCA and SpaRSA showed the best results over all. Even at very high compression rate, SpaRSA performed at about 20% better than other techniques for both PSVM and SVM.

5.2 SVM and SVR on SVR Data set

The SVR data proved to be a more difficult to perform classification on as it is hard to create boundaries to separate the classes in Figure 4. Figure 8 shows that, a correct rate of about 80% was the best correct rate and it was botained with the SVM classifier. For this data set, all of our compression techniques performed roughly the same. At high compression (compressed size less than 5 dimensions out of 30), PLS was able to maintain reasonable accuracy. We set up 4 SVM classifiers for 4-class classification. Each SVM only responsible for separating one class from the

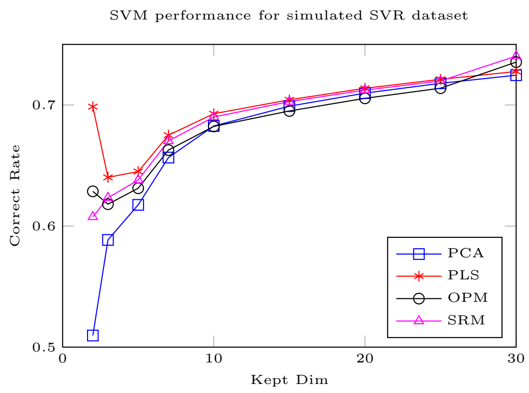

Figure 8: We created the SVR data set using 4 classes and applied 4 parallel SVM classifiers on the data. Each SVM determines whether each measurement belongs to one specific class or not. Based on this configuration, we can get 3 combinations. One is that for a data measurement, the result from the classifiers indicates that it belongs to only one class. Second is that the measurement belongs to multiple classes, similarly, the measurement could also be classified as it does not belong to any class. We looked at each SVM classifier's weight, and picked the class with the largest weight.

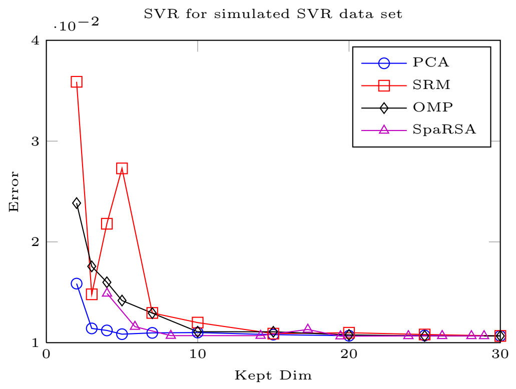

Figure 9: Performance of the SVR prediction on the SVR data set is very good. All the data reduction techniques had very low error rates, but PCA still stands out with its ability to reach the best prediction at any numbers of dimensions kept.

rest of the classes, the result is determined by combining the results from all the SVMs and their weights.

On the other hand, SVR prediction results were very encouraging. We only had less than 1.5% error rate throughout most of the simulation (Figure 9). At high compression (compressed size less than 5 dimensions), PCA approached the true value faster than all the other methods. As compressed size approached to 10 dimensions, all data reduction techniques had similar performance.

5.3 SVM and SVR on Turbofan Degradation Data Set

On the Turbofan data set, we used the one vs all method for multi-SVM classification, i.e. performing SVM for each class and combining the results at the end based on the weight of each classifier. For the 2-class SVM, we separated the training data based on the average of the system health. For the 4-class SVM, we picked 3 different ways to define our classes shown in Table 2. The performance of the classifiers is affected by the way we define our classes for the training set. As we lower the range for class assignment 1, we got a very low accuracy result. On the other hand, when we increased the range of class assignment 1 to 150 hours, we got much better accuracy. However, in practices the user would probably prefer a system with the capability of determining a tighter range of RUL. For example, if there is a replacement required for engines with 30 hours of RUL or less, it is more efficient to apply a system which detects up to 50 hours, so the uncertainty (waste of resource) is within 20 hours. If a measurement falls under 50 hours mark, it is recommend to replace it. On the other hand, if our system can only detect up to 200 hours, if the measurement falls under 200 hours mark, the uncertainty would be much higher. A system detecting up to 30 hours would be the best.

All four compression techniques performed similarly. With 2-class SVM classification, PCA and PLS, the correct rate dropped as we decreased the compression rate from 15 to 20. We got better results in terms of correct rate with higher RUL boundary for first class (we had it at 75 increased to 150).

An interesting result is shown in Figure 11, where when we lowered the rate of compression, the correct rate is also decreased. SRM struggled the most out of all compression techniques.

To test SVR with the Turbofan data set, we computed a new metric to quantify the error rate, $\left(\frac{\hat{Y}}{Y} - 1\right)^2$, where $\hat{Y}$ is the predicted RUL and Y is the actual RUL of the engine. We also compared with [21], applied their mean error metric for data set 1, namely $ME = \frac{\sum_{k=1}^{N_s} |e_k^t|}{N_s}$, where ME is mean error, N_s is the number of measurements, $e_k^t = \hat{y}_k^t - y_k$, $\hat{y}_k$ and y_k are the predicted RUL and true RUL of k^{th} measurement respectively. Results are shown in Figures 12 and 13. For both metrics, it turns out PCA still performs the best out of all the compression techniques. In Figure 13, SVR reached its best prediction values around 10 dimensions, and performance decreased as we reduce the compression level.

Table 2: SVM class range assignments and performance

Classes			Figure 10	Figure 11
Class1	$RUL < mean(RUL)$	$RUL < 75$	$RUL < 125$	$RUL < 150$
Class2	$RUL > mean(RUL)$	$75 < RUL < 150$	$125 < RUL < 175$	$150 < RUL < 200$
Class3	N/A	$150 < RUL < 200$	$175 < RUL < 225$	$200 < RUL < 250$
Class4	N/A	$RUL > 200$	$RUL > 225$	$RUL > 250$
Correct Rate	70%	60%	80%	90%

6. CONCLUSION

In our study, we have implemented the PCA, PLS, SRM, OMP, and SpaRSA data compression techniques to reduce the dimension of several data sets. We combined these data compression techniques with SVM, PSVM and SVR to determine the class label, and respectively the remaining useful life of an equipment. In general, for the data sets we tested, we can compress to half of the data set dimension and still get reasonable results. Better results can also be obtained by implementing data filtering techniques to further reduce the measurement noise that is embedded in the data (e.g. median filter).

Throughout our tests, we determined that PCA has been performing the best compared to the other techniques. It also has a better processing speed comparing to SpaRSA and OMP.

Occasionally, processing more information does not lead to improved results. Therefore, it is crucial to filter out unwanted information using data reduction techniques. Considering the classifier and predictor performance and processed data size, it is recommendable to try to compress the data

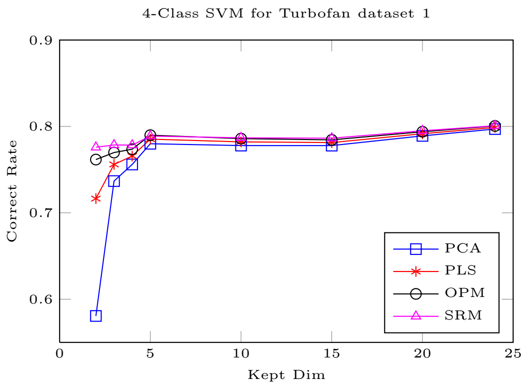

Figure 10: Measurement with the time to failure (TTF) time between 0 to 125 hours, we labeled as class 1. Similarly, class 2, 3 and 4 were determined by the TTF time range from 125 to 175 hours, 175 to 225 hours and 225 to 400 hours. With this configuration, we can compress the data down to 5 dimensions and still obtain a correct rate of 80%.

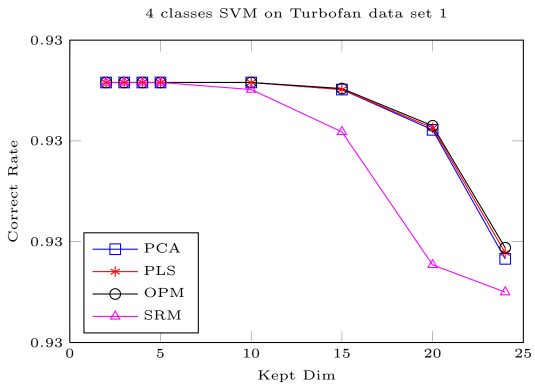

Figure 11: Measurements with the TTF time between 0 to 150 hours were labeled as class 1. Similarly, class 2, 3 and 4 were determined by the TTF time range from 150 to 200 hours, 200 to 250 hours and 250 to 400 hours. This configuration gives us the highest correct rate of 90%.

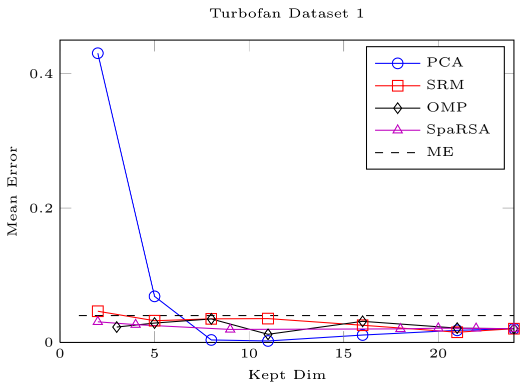

Figure 12: SVR performance on Turbofan data set 1 using the metric in [21] calculated with half of the engine units provided a .04 error rate using their learning technique, where our results are lower than .04 error mark for almost all of our techniques. The performance difference between different compressed dimensions is small. PCA is again the winer of this data set.

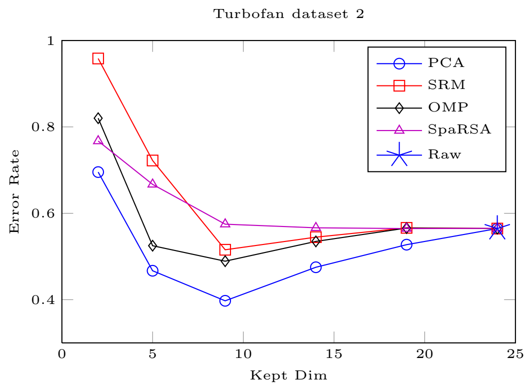

Figure 13: SVR performance on Turbofan data set 2 using half of the engine units from the data set in training and testing. The best compression is around 10 dimensions, which provides the lowest error rate for all the compression techniques. PCA proved to be the best compression technique for this data set.

before-hand, as the results of our tests suggest that we can save a lot of bandwidth and processing time while maintaining acceptable loss.

REFERENCES

[1] A. Asuncion and D. J. Newman, UCI Machine Learning Repository, University of California-Irvine, School of Information and Computer Science, 2007.

[2] *http://ti.arc.nasa.gov/project/prognostic-data-repository/*

[3] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation," *Proc. First Intl. Conf. on Prognostics and Health Management*, Denver, CO, Oct. 2008.

[4] H. Wold, "Path models with latent variables: The NIPALS approach," *Quantitative Sociology: International perspectives on mathematical and statistical model building*, pp. 307-357, Academic Press, 1975.

[5] H. Abdi, "Partial least square regression (PLS regression)," *Encyclopedia of Measurement and Statistics*, pp. 740–744, Sage Publications, 2007.

[6] S. de Jong, "SIMPLS: An alternative approach to partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, Vol. 18, pp. 251-263, Mar. 1993.

[7] R. Baraniuk, "Compressive sensing," *IEEE Signal Processing Magazine*, Vol. 24, No. 4, pp. 118-121, Jul. 2007.

[8] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, Vol. 52, No. 12, pp. 5406-5425, Feb. 2006.

[9] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, Vol. 52, No. 4, pp. 1289-1306, Apr. 2006.

[10] T. T. Do, L. Gan, Y. Chen, N. Nguyen, and T. D. Tran, "Fast and Efficient Dimensionality Reduction Using Structurally Random Matrices," *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Dallas, TX, Apr. 2009.

[11] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, Vol. 53, No. 12, pp. 4655-4666, Dec. 2007.

[12] W. Li and J. C. Preisig, "Estimation of Rapidly Time-Varying Sparse Channels," *IEEE Journal of Oceanic Engineering*, Vol. 32, No. 4, pp. 927–939, Oct. 2007.

[13] J. P. Lewis, "A Short SVM (Support Vector Machine) Tutorial," *www.idiom.com/∼zilla/code.html.*

[14] B. Scholkopf and A. J. Smola, "Learning with Kernels," *The MIT Press*, 2001.

[15] A. J. Smola and B. Scholkopf, "A Tutorial on Support Vector Regression," *http://www.springerlink.com/content/km7krm46802r2114/*, 1998.

[16] R. J. Georgescu, C. R. Berger, P. Willett, M. Azam and S. Ghoshal, "Comparison of Data Reduction Techniques Based on the Performance of SVM-type Classifiers", Aerospace Conference, Los Angeles, CA, Fab, 2010

[17] J. L. Williams, P. S. Maybeck,"Cost-Function-Based Gaussian Mixture Reduction for Target Tracking".

[18] J. L. Willams, "Gaussian Mixture Reduction for Tracking Multiple Maneuvering Targets in Clutter".

[19] A. Saxena and K. Goebel (2008). "Turbofan Engine Degradation Simulation Data Set", NASA Ames Prognostics Data Repository, NASA Ames, Moffett Field, CA.

[20] S. J. Weight, R. D. Nowake, M. T. Figueiredo, "Sparse Reconstruction by Separable Approximation".

[21] J. Sun, H. Zuo, H. Yang, M. Pecht, "Study of Ensemble Learning-Based Fusion Prognostics".

[22] L. I. Smith, "A Tutorial On Principal Components Analysis", *http://www.cs.otago.ac.nz/cosc453/student tutorials/principal components.pdf*, Feb 26, 2002.

[23] "Partial Least Squares (PLS)," *http://www.statsoft.com/textbook/partial-least-squares/.*

[24] M. Bayati, D. Shah, M. Sharma, "A Simpler Max-Product Maximum Weight Matching Algorithm and the Auction Algorithm", 2006.

[25] R. O. Duda, P. E. Hart, D. G. Stork, "Pattern Classification," *John Wiley and Sons, Inc*, pp. 259-264, 2001.

[26] K. R. Muller, A. Smola, G. Ratsch, B. Schololkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines", In B. Scholkopf, C.J.C. Burges, and A. J. Smola, editors, Advances in Kernel Methods Support Vector Learning, pp. 243C254, Cambridge, MA, 1999. *MIT Press. Short version appeared in ICANN97, Springer Lecture Notes in Computer Science.*

[27] S. R. Gunn, "Suport Vector Maichines for Classification and Regression", May, 1998

[28] A. J. Smola, B. Scholkophf, "A tutorial on Support Vector Regression", *NeuroCOLT2 Technica Report Series, NC2-TR-1998-03*, Oct, 1998.

[29] C. Chuang, C. Lin, "LIBSVM–A Library for Support Vector Machines", *http://www.csie.ntu.edu.tw/ cjlin/libsvm/matlab*, 2011.

[30] G. Fung, O. Mangasarian, "proximal Supoport Vector Machine Classifiers", University of Wisconsin, madison, WI, 2001.