14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

SUPERVISED COMPRESSION OF MULTIVARIATE TIME SERIES DATA

Victor Eruhimov, Vladimir Martyanov, Peter Raulefs and Eugene Tuv

Analysis & Control Technology, Intel,
5000 W Chandler Blvd, Chandler AZ85226, USA
victor.eruhimov@intel.com, vladimir.martyanov@intel.com, peter.raulefs@intel.com, eugene.tuv@intel.com

ABSTRACT

A problem of supervised learning from the multivariate time series (MTS) data where the target variable is potentially a highly complex function of MTS features is considered. This paper focuses on finding a compressed representation of MTS while preserving its predictive potential. Each time sequence is decomposed into Chebyshev polynomials, and the decomposition coefficients are used as predictors in a statistical learning model. The feature selection method capable of handling true multivariate effects is then applied to identify relevant Chebyshev features. MTS compression is achieved by keeping only those predictors that are pertinent to the response.

1. INTRODUCTION

The paper considers a problem of multivariate time series compression. We start with a dataset where each sample consists of several time series and a single response value. Individual series from the same sample correspond to the different variables with different physical characteristics generated by a process. Our goal is to reduce the representation size of time series, and at the same time to preserve the important information about the underlying process.

This work is motivated by the manufacturing applications where a set of sensors log down various physical properties of a wafer processing over time. The number of time series varies from several dozens up to hundreds (an example is a set of Fourier coefficients of 2MHz signal averaged over a window of 1 second). After a manufacturing step a wafer undergoes a so-called metrology, where the process quality is verified by taking additional measurements from the wafer. One of our key objectives is predicting the performance of the tool given sensors data. We achieve this by building a supervised statistical model from historical data to predict the metrology result. The quality of the model is assessed by predicting response on a set-aside portion of the dataset not used for learning, and calculating a prediction error by comparing the model output with the actual metrology result. The amount of raw data is much larger than what we can store, but potentially could be very useful in predicting the process performance. In this context we are interested in reducing the representation of time series data without significantly altering the quality of the prediction, we are not concerned here with the original signal reconstruction. This is why the approach taken here is different from the classical signal compression techniques. In order to stress this distinction we will refer to this problem as supervised signal compression.

Our methodology combines methods of signal processing and learning theory. First, we extract a set of features from each of the time series that is sufficient or superfluous for the original series reconstruction. Then we use our generic multivariate feature selection mechanism that filters out irrelevant and ranks informative for the response features. Given our requirements on the data compression rate we choose a number of the most important features and rebuild the model. The information loss rate is measured by the change in the estimate of the prediction error from the model built using all relevant features. We can balance between the compression rate and the predictive power of the model by changing the number of of the most important features used for the model construction.

The outline of the paper is as follows. Section 2 focuses on the features that we extract from time series, Sections 3 and 4 provide the background on the supervised learning techniques, variable selection and ranking methods that we used, Sections 5 and 6 describe the experimental setup, and the discussion of the results.

2. CHEBYSHEV POLYNOMIALS

A variety of feature representations are used in practical applications for learning time series models. More common are Principal Components, Fourier [4], and wavelet coefficients [8]. In this paper we use Chebyshev polynomials expansion to represent the time series. The problem of choosing a feature basis set is out of the scope of this paper, and we will not go beyond noting that in our experiments Chebyshev features tested against other more common indexing techniques on various datasets demonstrated comparable or better performance.

Chebyshev polynomial (see [12] for a brief overview and references) $y(x) = T_n(x)$ of degree n by definition is a polynomial solution of the equation

$$(1-x^2)\frac{d^2y}{dx^2} - x\frac{dy}{dx} + n^2y = 0, \tag{1}$$

where $|x| \leq 1$ and n is a non-negative integer. For $n = 0$ $T_0(x) = 1$. Chebyshev polynomials can also be calculated using one of useful properties:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \tag{2}$$

$$T_n(x) = cos(n \cdot cos^{-1}(x)). \tag{3}$$

A set of Chebyshev polynomials $\{T_n(x)\}_{n=0,1,...}$ is orthogonal with respect to the weighting function $(1-x^2)^{-1/2}$:

$$\int_{-1}^{1} \frac{T_m(x)T_n(x)dx}{\sqrt{1-x^2}} = \left\{ \begin{array}{l} \frac{1}{2}\pi\delta_{nm}, n>0, m>0 \\ \pi, n=0, m=0 \end{array} \right. , \tag{4}$$

where δ_{mn} is the Kronecker delta.

Using the last property we can represent any piecewise continuous function $f(x)$ in the interval $-1 \leq x \leq 1$ as a linear combination of Chebyshev polynomials:

$$\sum_{0}^{\infty} C_n T_n(x) = \begin{cases} f(x), \text{where } f(x) \text{ is continuous} \\ \frac{f(x-0)+f(x+0)}{2} \text{ in discontinuity points} \end{cases} . \quad (5)$$

Here

$$C_n = \frac{A}{\pi} \int_{-1}^{1} \frac{f(x)T_n(x)dx}{\sqrt{1-x^2}}, \qquad A = \begin{cases} 1, n = 0 \\ 2, n > 0 \end{cases} . \quad (6)$$

For a function $\{f_i\}_{i=1,\ldots,P}$ defined on a discrete domain we calculate the coefficients of the Chebyshev decomposition using a straightforward formula:

$$C_n = \frac{A}{\pi} \sum_{i=1}^{P} \frac{f_i T_n(x_i)}{\sqrt{1-x_i^2}}, \quad (7)$$

where $x_i = -1 + \frac{2}{P}(i - \frac{1}{2})$.

Generally we start with a set $\{C_n\}_{n=0,\ldots,P-1}$ of size P and then use a supervised learning technique to remove redundant features.

3. TREE ENSEMBLES

This section gives background on supervised learning techniques that we employ both for the feature selection and for the prediction. We try to address a problem of feature filtering, or removal of irrelevant inputs in a very general supervised setting: the target variable could be numeric or categorical, the input space could have variables of mixed type with non-randomly missing values, the underlying relationship of response Y and predictors X could be very complex with nontrivial interactions. The data could be massive in both dimensions (tens of thousands of variables, and millions of observations). Ensembles of unstable but very fast and flexible base learners such as trees (with embedded feature weighting) can address the most of the listed challenges. They have been proven to be very effective for variable ranking in problems with up to a hundred thousand predictors [1, 9]. A more comprehensive overview of feature selection with ensembles is given in [10].

A decision tree partitions the input space into a set of disjoint regions, and assigns a response value to each corresponding region. It uses a greedy, top-down recursive partitioning strategy. At every step a decision tree uses exhaustive search by trying all combinations of variables and split points to achieve the maximum reduction in impurity of the node. Therefore, the tree constructing process itself can be considered as a method of variable selection, and the impurity reduction due to a split on a specific variable could indicate the relative importance of that variable to the tree model. Note, that this relative importance is based on a multivariate model, and it is different from the relevance measured by standard, univariate filter methods. For a single decision tree, a measure of variable importance is proposed in [3]:

$$VI(x_i, T) = \sum_{t \in T} \Delta I(x_i, t) \quad (8)$$

where $\Delta I(x_i, t) = I(t) - p_L I(t_L) - p_R I(t_R)$ is the decrease in impurity due to an actual (or potential) split on variable x_i at a node t of the optimally pruned tree T. The sum in (8) is taken over all internal tree nodes where x_i is a primary splitter. Node impurity $I(t)$ for regression is defined as $\frac{1}{N(t)} \sum_{s \in t} (y_s - \bar{y})^2$ where the mean $\bar{y}$ and sum are taken over all observations of the response y_s in node t, and $N(t)$ is the number of observations in node t. For classification $I(t) = Gini(t)$ where $Gini(t)$ is the Gini index of node t:

$$Gini(t) = \sum_{i \neq j} p_i^t p_j^t \quad (9)$$

and p_i^t is the proportion of observations in t whose response label equals i $(y = i)$ and i and j run through all response class numbers. The Gini index is zero when t has observations only from one class, and reaches its maximum when the classes are perfectly mixed.

One of the most recent advances in tree ensembles - GBT (gradient tree boosting) [5, 6] has been proven to be among the most accurate and versatile state-of-the-art learning machines. GBT is a serial ensemble where every new tree constructed relies on previously built trees. At every iteration l of GBT a new tree T_l is fitted to the generalized residuals with respect to a loss function Ψ

$$-\left[\frac{\partial \Psi(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{l-1}} \quad (10)$$

giving terminal regions $R_{jl}, j = 1, 2, \ldots, J_l$. The corresponding constants γ_{jl} are solutions

$$\gamma_{jl} = \arg\min_{\gamma} \sum_{x_i \in R_{jl}} \Psi(y_i, F_{l-1}(x_i) + \gamma) \quad (11)$$

and

$$F_l(\mathbf{x}) = F_{l-1}(\mathbf{x}) + \nu \cdot \sum_{j=1}^{J_l} \gamma_{jl} I(\mathbf{x} \in R_{jl}) \quad (12)$$

where $0 < \nu < 1$ is a regularization parameter (learning rate.) The solution is given by

$$\hat{F}(\mathbf{x}) = F_L(\mathbf{x}), \quad (13)$$

where the size of the ensemble L is chosen to avoid overfitting (usually by monitoring validation errors.)

GBT inherits all nice properties of a single tree, and also provides (as a byproduct) more reliable estimate of the variable importance. The importance measure (8) is averaged over the trees in the ensemble

$$VI(x_i) = \frac{1}{L} \sum_{l=1}^{L} VI(x_i, T_l) \quad (14)$$

GBT builds shallow trees using all variables (on a subsample of the training data), and hence, it can handle large datasets with a moderate number of inputs. Very high dimensional data (thousands or even several hundreds of features) is extremely challenging for GBT. A modification of GBT [1] suggests a different ensemble learning strategy so that processing of very high dimensional datasets is feasible with almost no loss in prediction accuracy.

Random Forest [2] is a distinguished representative of tree ensembles that extends the "random subspace" method [7]. It grows a forest of random trees on bagged samples showing excellent results comparable with the best known classifiers. Random Forest (RF) does not overfit, and can be summarized as follows:

1. a number n is specified much smaller than the total number of variables N (typically $n \sim \sqrt{N}$)
2. each tree of maximum depth is grown on a bootstrap sample of the training set
3. at each node, n out of the N variables are selected at random
4. the split used is the best split on these n variables

The computational complexity for each tree in the RF $\sim \sqrt{N}\ S\ log(S)$, where S is the number of the training cases. Therefore, it can handle very large number of variables with moderate number of observations.
Note that for every tree grown in RF, about one-third of the cases are out-of-bag (out of the bootstrap sample). The out-of-bag (OOB) samples can serve as a test set for the tree grown on the non-OOB data.

The variable importance for RF can be defined as for GBT (14) by averaging the importances from individual trees.

We use GBT ensemble for response prediction since our response is a numeric variable and GBT is natively better adjusted for the regression task given the sample size is not too small. At the same time Random Forest is used in the process of feature selection for performance reasons. It is important to note that we could use any of these methods for both tasks.

4. ENSEMBLE BASED FEATURE RANKING AGAINST ARTIFICIAL CONTRASTS

Relative feature ranking (14) provided by the ensembles mentioned above, does not separate relevant features from irrelevant. Only a list of importance values is produced without a clear indication which variables to include, which to discard. Also, trees tend to split on variables with more distinct values. This effect is more pronounced for categorical predictors with many levels. It often makes a less relevant (or completely irrelevant) input variable more “attractive” to split on only because it has high cardinality.

The main idea in [11] relies on the following reasonable assumption: a stable feature ranking method, such as an ensemble of trees, that measures relative relevance of an input to a target variable Y would assign a significantly (in statistical sense) higher rank to an important variable x_i than to an artificial variable created from the same distribution as x_i, independently of Y. Here we give a brief description of the algorithm described in [11] that we used as a basis for our experiments.

The method is a combination of three ideas: **A)** Estimating importance using RF ensemble of trees of fixed depth (3-6 levels) with the split weight re-estimation on OOB samples (gives more accurate and unbiased estimate of variable importance in each tree and filters out noise variables), **B)** comparing variable importance against artificially constructed noise variables using a formal statistical test, and **C)** iteratively removing the effect of identified important variables to allow detection of less important ones (trees and parallel ensemble of trees are not well suited for additive models).

4.0.1 A. Split weight re-estimation.

A modified scheme for calculating split weight and selecting best split in each node of a tree is proposed. The idea is to use training samples to find best split point on each variable, then use samples that were not used for building the tree (out-of-bag), to select best split variable in a node. Split weight used for variable importance estimation is also calculated using out-of-bag samples.

4.0.2 B. Selecting important features.

In order to determine a cut-off point for the importance scores, there needs to be a *contrast* variable that is known to be truly independent of the target. By comparing variable importance to this contrast (or several), one can then use a statistical test to determine which variables are truly important. We propose to obtain these artificial contrast variables by randomly permuting values of original N variables across the M examples. Generating contrasts using unrelated distributions, such as Gaussian or uniform, is not sufficient, because the values of original variables may exhibit some special structure.

Trees in ensemble are then broken into K short sets of equal size $J = 10 - 50$. Variable importance is then computed for all variables, including the artificial contrasts for each set. Using sets is important when the number of variables is large or tree depth is small, because some (even important) features can be absent in a particular tree. To gain statistical significance, importance score of all variables is compared to a percentile (we used 75^{th}) of importance scores of the N contrasts. A statistical test (Student’s t-test) is performed to compare the scores over K series. Variables that are scored significantly higher than contrasts are selected.

4.0.3 C. Removing effects of identified important variables.

After a subset of relevant variables were discovered by the step B, we need to remove their effects on the response. To accomplish this, the response is predicted using only these important variables, and a residual of the response is computed. Then we return to the step A, until no variables remain with scores significantly higher than those of the contrasts. It is important that the step A uses all variables to build the ensemble, and does not exclude identified important ones.

Clearly we could use the described feature selection scheme with any classifier/regressor function which provides variable importance from all variable interactions. To our knowledge, only ensembles of trees can provide this conveniently.

5. DATA GENERATION

Due to confidentiality of the manufacturing process information we used a data generator designed specifically to mimic most of the challenges we face in the real environment. Each sample in the dataset consisting of several time series and the response value is generated using the following algorithm. First, we generate $2N$ time series independently from each other. We start with a trapezoid given by vertices randomly positioned in a predefined region (see Figure 1 (a)). Then we add random amounts of curvature given by parameters v_l and v_r into the beginning and ending stages of the trapezoid respectively. Harmonic variation of random frequency, phase and Gaussian-modulated amplitude, given by parameter v_a, are added to the middle part of the signal. Values of v_l, v_r and v_a are sampled from a predefined distribution. The resulting function is evaluated at P equidistant points and about 1% of Gaussian noise is added. An example of a final signal is presented in Figure 1 (b). Then we use first N time series to generate the response y by computing a sum of a linear and quadratic forms from time series parameters $\{v_l, v_r, v_a\}$:

$$y = AV + V^T BV + \varepsilon. \qquad (15)$$

Here A is a vector $1 \times N$, B is a matrix $N \times N$ and ε is Gaussian noise. Elements of A and B are samples from a predefined distribution function and are the same for the whole dataset. The other N time series are included to check the robustness of the learning and compression engines to the noise. Following an analogy with real data we refer to the time series with number i ($1 \le i \le 2N$) as sensor i.

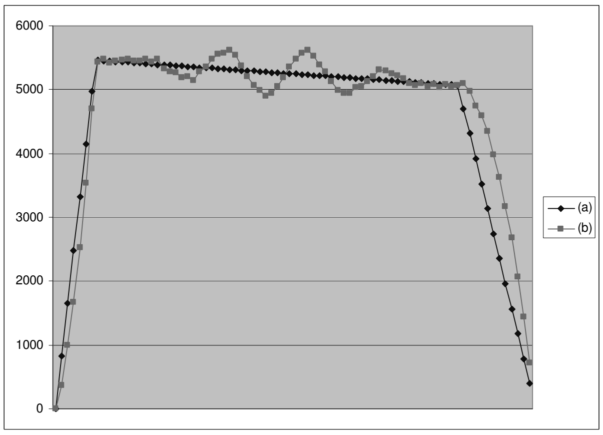

Figure 1: An example of generated time series. (a) Trapezoid with random vertices; (b) Final signal with oscillation and noise.

In the experiments we purposely used large signal to noise ratio. It would correspond to the most conservative scenario in terms of data compression since the prediction error degradation will be most noticeable with exclusion of even weak features-predictors.

6. EXPERIMENTAL RESULTS

For the experiments we used a dataset with 20 sensors and 3000 samples created by the data generator described in the previous section. Each time series contained measurements at 80 time points. We extracted 80 Chebyshev coefficients from it (this corresponds to a lossless compression), but due to the nature of the data we used only first 25 of them. The resulting dataset had $25 \cdot 20 = 500$ predictors. First, we built a GBT model on the full dataset to estimate the prediction error. Then, we ran a feature selection algorithm [11] that filters out irrelevant features, and ranks predictors according to their predictive power. Next, we re-built the model using several most important predictors. Consistently throughout the experiments we used 70% of the samples for model construction, and the rest 30% for the prediction error estimation.

Figure 2 shows the dependence of the prediction error on the number of features selected. The error graph is normalized by the standard deviation of the response (the prediction error of the trivial model - overall mean). The dotted line indicates the level of prediction error when all 500 features are used – this is the lowest error we can get given GBT is not sensitive to the noise variables. One can see that the number of features could be decreased by an order of magnitude with the *relative* prediction error increase of about 20%.

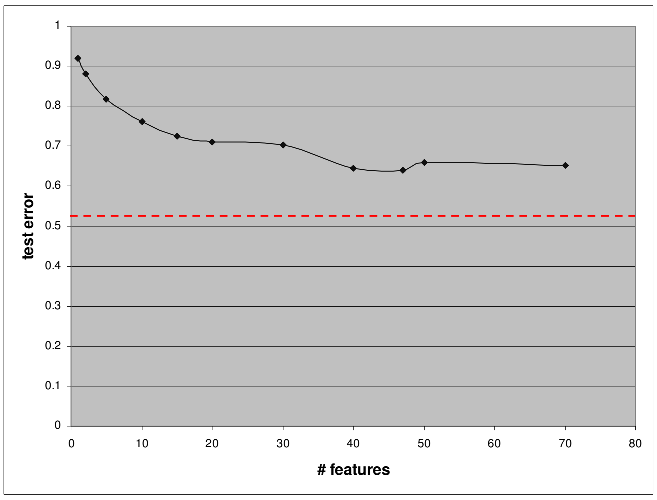

Figure 2: Test error vs. the number of the most important features used for prediction.

Figure 3 shows the features that are selected as the most important. The first number in the feature name indicates the sensor and the second – the Chebyshev coefficient. Note that first 20 features come from only 5 sensors out of 20. Figure 4 demonstrates the importance of sensors. The first series in the chart corresponds to sensor importance defined as the maximum estimated by the algorithm importance of the features derived from that sensor.The second series is the “true” sensor importance that is calculated from matrices A and B used to generate the data. The importances for sensors 11 to 20 are zero because these series were not participating in response generation, and the feature selection algorithm captured that. Note that 20 features with the highest importance (see Figure 3) came from the 5 sensors with the top “true” importances (i.e. sensors that have the largest coefficients in A and B used to generate the response).

Figure 5 shows the original time series overlayed with two its reconstructions from Chebyshev coefficients selected as most important for prediction (top two, and top eight coefficients). The signal reconstructed from two coefficients is shifted down – zero coefficient corresponding to the series mean was not used in the reconstruction.

7. CONCLUSION

We considered a problem of multivariate time series compression in the supervised setting. Given a response variable we are interested only in informative (in terms of prediction) features extracted from the multiple time sequences. Non-informative (or less informative) features are disregarded - compressed out. The amount of compression is balanced with potential loss of the prediction accuracy. The proposed method uses an efficient basis function decomposition for each time series, followed by an automatic and truly multivariate (any level of interactions) feature selection and ranking mechanism that filters out irrelevant features (expansion coefficients), and ranks important ones with respect to the response.

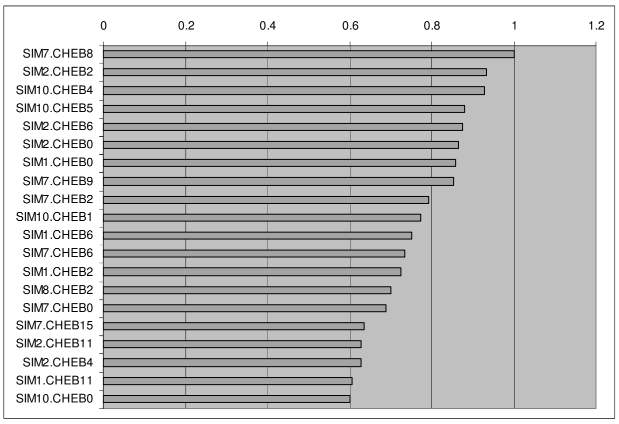

Figure 3: A list of features together with their importance scores in descending order. The feature SIM$\langle i\rangle$.CHEB$\langle j\rangle$ corresponds to the j-th coefficient of Chebyshev decomposition of i-th time series.

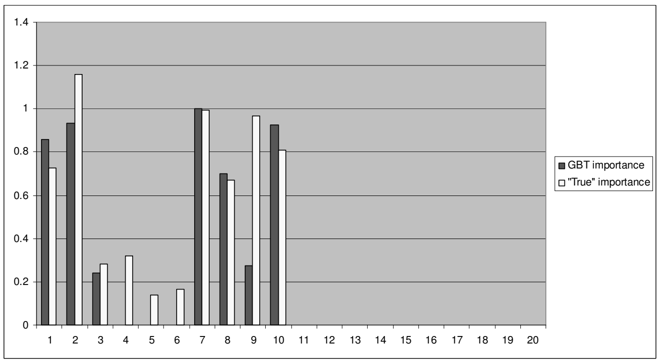

Figure 4: Importance of sensors calculated by the GBT model and the data generator.

The experimental setup corresponded to the challenges high precision semiconductor manufacturing faces routinely. The proposed approach was illustrated by representing individual time sequences using coefficients in Chebyshev polynomials decomposition (could be any common time series decomposition). We showed that at least an order of magnitude data compression rate could be achieved with insignificant loss of prediction accuracy under a conservative signal-to-noise assumption.

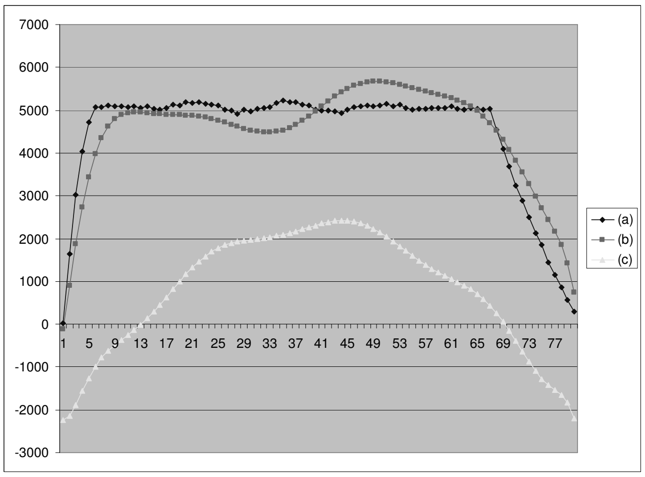

Figure 5: An example of (a) original time series; (b) its reconstruction from 8 Chebyshev coefficients (number 0,2,4,5,6,8,11,15) suggested as the most important for prediction by GBT; (c) same as (b) for 2 coefficients (number 2 and 11).

REFERENCES

[1] A. Borisov, V. Eruhimov, and E. Tuv. Dynamic soft feature selection for tree-based ensembles. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, New York, 2006.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. CRC Press, 1984.

[4] Chris Chatfield. *The Analysis of Time Series*. Chapman & HALL/CRC, 2004.

[5] J.H. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University, 1999.

[6] J.H. Friedman. Stochastic gradient boosting. Technical report, Dept. of Statistics, Stanford University, 1999.

[7] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[8] Stephane Mallat. *A Wavelet Tour on Signal Processing*. Academic Press, 1999.

[9] Kari Torkkola and Eugene Tuv. Ensembles of regularized least squares classifiers for high-dimensional problems. In Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, 2006.

[10] E. Tuv. Feature selection and ensemble learning. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, New York, 2006.

[11] E. Tuv and K. Torkkola. Feature filtering with ensembles using artificial contrasts. *IEEE Intelligent Systems Journal, November/December*, 2005.

[12] Eric W. Weisstein. Chebyshev polynomial of the first kind. *From MathWorld–A Wolfram Web Resource*.