Weighted Order Statistic Classifiers with Large Rank-Order Margin

Reid Porter RPORTER@LANL.GOV
Damian Eads EADS@LANL.GOV
Don Hush DHUSH@LANL.GOV
James Theiler JT@LANL.GOV
Los Alamos National Laboratory, MS D440, Los Alamos, NM 87545 USA

Abstract

We investigate how stack filter function classes like weighted order statistics can be applied to classification problems. This leads to a new design criteria for linear classifiers when inputs are binary-valued and weights are positive. We present a rank-based measure of margin that is directly optimized as a standard linear program and investigate its relationship to regularization. Our approach can robustly combine large numbers of base hypothesis and has similar performance to other types of regularization.

1. Introduction

We investigate the two-class classification problem when the classifier is a weighted order statistic. In two-class classification we find a function $f(\vec{x}) : \mathrm{R}^D \to \{-1, 1\}$ that minimizes the probability of misclassification defined on a distribution G over $\mathrm{R}^D \times \{-1, 1\}$:

$$Pr_G[yf(\vec{x}) < 0] \tag{1}$$

This probability is called generalization error and is usually approximated with a loss function based on a training set. A training set is a sample

$$S = \langle(\vec{x}(1), y(1)), (\vec{x}(2), y(2)), \ldots, (\vec{x}(N), y(N))\rangle \tag{2}$$

where examples are chosen independently and at random from the distribution G. One approximation that is often used is misclassification error on the training set:

$$Pr_S[yf(\vec{x}) < 0]. \tag{3}$$

Recently, theoretical and practical results have suggested loss functions that consider sample margins sometimes improve approximation to generalization error (Mason et al., 2000). In this case, the loss function also includes the distance of each sample to the decision boundary, $yf(x)$. A number of theorems from the probably approximately correct machine learning framework bound (1) in terms of sample margins, for a number of function classes. These theorems generally take the following form (Schapire et al., 1998):

Theorem: For $\delta > 0$ with probability $1 - \delta$ over the random choice of the training set S, every function $f(\vec{x})$ in a function class $\mathcal{F}$ satisfies the following bound for all $\gamma > 0$:

$$Pr_G[yf(\vec{x}) \leq 0] \leq Pr_S[yf(\vec{x}) \leq \gamma] + \epsilon(\delta, N, |\mathcal{F}|, \gamma). \tag{4}$$

In this inequality, generalization error Pr_G is bound by training error Pr_S at margin γ and a second term ϵ that depends, among other things, on a complexity measure for the function class $\mathcal{F}$, $|\mathcal{F}|$. For a number of function classes, an increase in γ reduces ϵ by decreasing the complexity term $|\mathcal{F}|$. In practice, increasing γ will also increase Pr_S and the appropriate trade-off must be chosen by the learning algorithm. Several learning algorithms parameterize the tradeoff with a single parameter e.g. introducing a regularization term into the loss function (3).

Theorems that characterize the precise relationship (4) are developed for specific function classes $\mathcal{F}$ e.g. linear classifiers and other neural networks. In this paper, we suggest a similar relationship exists for weighted order statistic, and other stack filter derived function classes.

We present a brief account of weighted order statistic literature in Section 2. In Section 3, we show how weighted order statistic classifiers reduce to binary domain classifiers due to thresholding. This links the weighted order statistic classifier to other binary domain classifiers that are associated most often with ensemble methods.

We will propose a new type of margin for these function classes called rank-order margin. Our results apply to a large number of ordered function classes including all digital mappings $\{0,1\}^D \rightarrow \{0,1\}$ (Lin & Coyle, 1990). In Section 4, we investigate the relationship between rank-order margin, regularization and generalization with two synthetic experiments. We compare our approach to boosting with real-world data sets from the UCI machine learning repository. We summarize and discuss future research efforts in Section 5.

2. Background

Generalized median or weighted order statistic (WOS) filters are an active research topic in non-linear digital filtering (Astola & Kuosmanen, 1997). Their robustness to impulsive noise has made them a popular alternative in applications where linear filters perform poorly. The linear classifier, or thresholded linear filter, is a direct generalization of the weighted average and sample mean. In a similar way, the median filter is generalized to weighted median and the WOS filter. A more detailed analogy was presented by Arce (1998). In this paper we consider WOS classifiers, or thresholded WOS filters.

Our analysis and method of optimization are based on techniques known as threshold decomposition and stacking. These techniques were suggested for analysis and design of non-linear digital filtering. Our novel contribution is to apply these techniques to large margin loss functions to solve classification problems.

Our approach was inspired by Preston (1990), who described thresholded rank-order filters for a target detection problem. Several other classifiers are related to WOS classifiers, namely min-max classifiers (Yang & Maragos, 1995), and morphological networks (Ritter & Sussner, 1996).

WOS classifiers also relate to binary domain classifiers used in ensemble methods. The median, or majority vote is used by bagging (Breiman, 1996). A weighted order statistic is used to combine weak hypothesis in the adaboost m.1 algorithm (Schapire et al., 1998). Direct optimization of margin for this function class was suggested by Mason et al. (2000) and finding the maximum, minimum margin solution was considered by Grove and Schuurmans (1998). Tumer and Ghosh (1999) investigated the performance of order statistics for combining ensembles. In this paper we consider the weighted order statistic, and stack filter combiners and suggest a mechanism to control complexity.

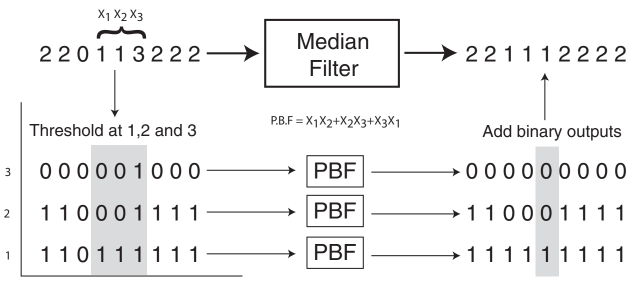

Figure 1. Threshold decomposition architecture for a three-input median filter

2.1. Threshold Decomposition

Threshold decomposition is an important tool for the analysis and design of non-linear digital filters (Fitch et al., 1984). We describe threshold decomposition for D inputs: $\vec{x} = [x_1, x_2, \ldots, x_D]$ where each input takes on values from a finite set of quantization levels: $x_i \in \{-q+1, \ldots, -1, 0, 1, \ldots, q\}$. Threshold decomposition was presented for real-valued inputs by Yin and Neuvo (1994) and Arce (1998). It is convenient to define a threshold function based on the indicator function:

$$T(\vec{x}, t) = [I(x_1 \geq t), I(x_2 \geq t), \ldots, I(x_D \geq t)] \tag{5}$$

Threshold decomposition applies the threshold function to the inputs at each quantization level. The original inputs can be exactly recovered from this decomposition by

$$x_i = -q + \sum_{t=-q+1}^{q} T_i(\vec{x}, t) \tag{6}$$

where T_i is the i'th component of the vector function output (5).

2.2. Stack Filters

If we define a binary function $f(\vec{u}) : \{0,1\}^D \rightarrow \{0,1\}$ and apply it to each level of the threshold decomposition then the function:

$$S_f(\vec{x}) = -q + \sum_{t=-q+1}^{q} f(T(\vec{x}, t)) \tag{7}$$

defines a stack filter when $f(\vec{u})$ satisfies the stacking constraints. The stacking constraints are:

$$f(\vec{u}) \leq f(\vec{v}) \text{ whenever } u_i \leq v_i \text{ for all } i. \tag{8}$$

A necessary and sufficient condition for $f(\vec{u})$ to satisfy the stacking constraints is that it is a Positive Boolean Function (PBF) (Wendt et al., 1986). PBFs are the subset of Boolean functions that can be expressed without complements of the input variables. Figure 1 illustrates the stack filter architecture, and corresponding PBF for a three input median function.

2.3. Weighted Order Statistics

Weighted order statistic filters are a useful sub-class of stack filters. In this case, $f(\vec{u})$ is restricted to a linearly separable positive boolean function:

$$f(\vec{u}) = I(\vec{w}\vec{u}^T \geq b) \tag{9}$$

where $w_i, b \in \{0\} \cup \mathrm{R}^+$ for all i.

By choosing weights and threshold b, median, order statistic, weighted median and weighted order statistic function classes can be implemented (Yli-Harja et al., 1991). In figure 1, the linear seperable PBF for a 3 input median function has all $w_i = 1$ and $b = 1.5$. WOS functions, like the median, can be implemented directly in the real domain with sorting (Arce, 1998). However when targeting specialized hardware, like Field Programmable Gate Arrays, extremely efficient implementations of the threshold decomposition architecture are possible (Chen, 1989).

2.4. Data Dependent Threshold Decomposition

We have described threshold decomposition where inputs take on quantization values from a finite set. A useful property of stack filters is that the filter output is always equal to one of the filter inputs. This means a more general decomposition can be defined that can be applied to inputs with arbitrary range and precision. We first define some new notation. For an input vector $\vec{x} = [x_1, x_2, \ldots, x_D]$ where $x_i \in \mathrm{R}$, we define

$$x_{(i)} = i^{\text{th}} \text{ smallest value in } \vec{x} \tag{10}$$

For example, sorting elements of the vector $\vec{x}$ into ascending order would produce a vector:

$$[x_{(1)}, x_{(2)}, \ldots, x_{(D)}] \tag{11}$$

Let $SI_f(\vec{x})$ be a stack filter defined as:

$$SI_f(\vec{x}) = \sum_{t=1}^{D} f(T(\vec{x}, x_{(D)})) \tag{12}$$

There is a one-to-one correspondence between a stack index filter and a stack filter:

$$S_f(\vec{x}) = x_{(SI_f(\vec{x}))}. \tag{13}$$

3. Stack Filter and WOS Classifiers

We threshold the stack filter at 0 to produce a stack filter classifier which outputs class labels $y \in \{0, 1\}$:

$$y = I(S_f(\vec{x}) \geq 0). \tag{14}$$

Since the stack filter output is always one of the inputs, if $x_i >= 0$ for all i then the stack filter must assign the sample to class $y = 1$. This may not be a limitation in some applications, however in other cases, a more flexible method of combination is required e.g. inputs correspond to randomly generated features. One simple way to increase the model complexity is to supplement the stack filter inputs with mirrored input samples:

$$\vec{x} \leftarrow [x_1, x_2, \ldots, x_D, -x_1, -x_2, \ldots, -x_D]. \tag{15}$$

For the rest of this paper we assume a mirrored input space is used. This approach was suggested in the context of a mirrored threshold decomposition architecture by Paredes and Arce (2001). For classification, the mirrored input space provides an absolute reference point for the stack filter. That is, the median filter, when applied to the mirrored input space augmented with 0, will always output 0 (the additional 0 is not necessary in practice). An alternative interpretation, suggested by the weak-ordering property of stack filters around the median (Lin et al., 1994), is that the hard decision boundary at 0 has been replaced with a relative boundary defined by the median.

We reformulate the stack filter classifier (14) using the stack index filter definition (12). Since zero is guaranteed to lie between $x_{(D)}$ and $x_{(D+1)}$ we have:

$$y = I\left(\sum_{t=1}^{2D} f\left(T(\vec{x}, x_{(t)})\right) \geq D + 1\right) \tag{16}$$

and applying the stacking property,

$$y = f(T(\vec{x}, 0)). \tag{17}$$

Observe in (17) that the stack filter classifier reduces to a binary domain classifier, which is applied to inputs thresholded at zero. This is an important observation since it links stack filter classifiers to other techniques in machine learning that build binary domain classifiers. It also provides some insight into the complexity of stack filter classifiers compared to more popular function classes e.g. the WOS classifier is equivalent to thresholding all inputs at zero and then applying a linear classifier.

3.1. Rank-Order Margin

In this section we apply the concept of margin to stack filter and WOS classifiers. It is convenient to return to

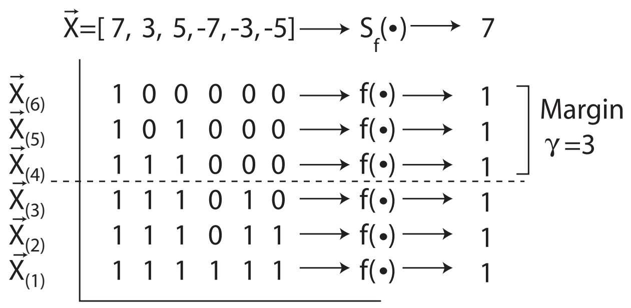

Figure 2. Example of rank-order margin.

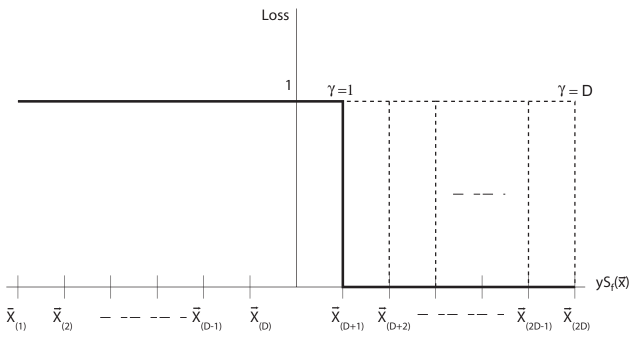

Figure 3. Rank-order margin loss functions.

class labels $y \in \{-1, 1\}$. The stack filter output is one of the input samples, and therefore sample margins $yS_f(\vec{x})$ take values from a finite, discrete set:

$$yS_f(\vec{x}) \in \{x_1, x_2, \ldots, x_D, -x_1, -x_2, \ldots, -x_D\}. \quad (18)$$

The classifier decision boundary is at 0, and we measure the distance to the decision boundary by rank order. Figure 2 is an example of the rank-order margin in the context of the threshold decomposition architecture. The mirrored input sample $\vec{x}$ belongs to class $y = 1$. A particular stack filter $S_f(\vec{x})$ produces a correct classification $S_f(\vec{x}) = 7$ with rank-order margin $\gamma = 3$.

3.2. Loss Functions

We now write a loss function for the stack index filter that includes rank-order margin. Training error (3) follows from (17):

$$\begin{aligned} Pr_s[(f(T(\vec{x}, 0)) \neq 1, y = 1) \cup \\ (f(T(\vec{x}, 0)) \neq 0, y = -1)]. \end{aligned} \quad (19)$$

The stacking property means training error at rank-order margin γ can be defined as:

$$\begin{aligned} Pr_s[(f(T(\vec{x}, x_{(D+\gamma)})) \neq 1, y = 1) \cup \\ (f(T(\vec{x}, x_{(D+2-\gamma)})) \neq 0, y = -1)] \end{aligned} \quad (20)$$

where $\gamma \in [1, 2, \ldots, D]$. The rank-order loss functions are illustrated in Figure 3.

3.3. Optimization

The loss function (20) applies to the stack filter function class. Wendt et al. (1986) show that under the mean absolute error criteria, stack filter optimization can be posed as a linear programming problem. The same linear program can be applied to the loss function (20) however the number of variables in the linear program grows exponentially with the input dimension. For the remainder of this paper we investigate the WOS classifier subclass where the number of variables is linear with input dimension.

Optimizing (20) for the WOS subclass (9) is a linear classifier design problem, subject to the stacking constraints: $\vec{w}, b \geq 0$. We approximate misclassification error with the perceptron criteria and then use a linear or quadratic program, depending on the type of regularization that we choose.

Training samples for the linear classifier are binary vectors that depend on the rank-order margin γ. For each pair $\vec{x}(n), y(n)$ in the training sample (2) we define:

$$\vec{v}(n) = \begin{cases} T\left(\vec{x}(n), x(n)_{(D+\gamma)}\right) & \text{when } y(n) = 1 \\ T\left(\vec{x}(n), x(n)_{(D+2-\gamma)}\right) & \text{otherwise} \end{cases} \quad (21)$$

We augment and transform the binary domain training set and weight vector:

$$\vec{z}(n) = [y(n)\vec{v}(n)|y(n)] \quad (22)$$
$$\vec{w} = [w_1 \ldots w_{2D}|b]. \quad (23)$$

The optimization problem is:

$$\begin{gathered} \min_{\vec{\tau}, \vec{w}} C \sum_{n=1}^{N} \tau_n + \Re(\vec{w}) \\ \text{subject to} \\ \tau_n \geq 1 - \vec{w}\vec{z}^T(n), \\ \vec{\tau} \geq 0, \text{ and } \vec{w} \geq 0 \end{gathered} \quad (24)$$

where $\Re(\vec{w}) = |\vec{w}|$ for L_1 regularization and the solution is found with a linear program, or $\Re(\vec{w}) = \|\vec{w}\|$ for L_2 regularization and the solution is found with a quadratic program.

We suggest using rank-order margin γ as a free parameter to control the complexity of WOS classifiers. In experiments we will fix the parameter C at 500 and then select the optimization problem by varying γ in (21). We compare this approach to regularization in which case we fix γ at 0 and then select the optimization problem by varying C in (24).

3.4. Expansion on Training Samples

For some classification problems the choice of regularization $\Re(\vec{w})$ can significantly affect classifier performance. We will see examples of this in our synthetic experiments. When using rank-order margin as the free parameter the choice of regularization may have little effect since C is fixed at 500. We used two optimization problems to investigate this further. The first optimization problem was presented in (24) where we solve for $\vec{w}, b$ and find a classifier of the form:

$$f(\vec{u}) = I(\vec{w}\vec{u}^T \geq b) \tag{25}$$

where $\vec{u} = T(\vec{x}, 0)$. We call this the primal problem.

We propose a second optimization problem called the expansion problem that explicitly expands $\vec{w}$ in terms of the training samples, and solves for new variables, $\vec{\alpha}$ and b:

$$f(\vec{u}) = I\left(\sum_{\{n|Y(n)=1\}} \alpha_n \left(\sum_{i=1}^{2D} u_i(n) \wedge u_i\right) + \sum_{\{m|Y(m)=-1\}} \alpha_m \left(\sum_{i=1}^{2D} u_i(m) \vee u_i\right) \geq b\right) \tag{26}$$

This expansion has two significant properties. First, the mapping is explicit and therefore when solving (24) in terms of $\vec{\alpha}$ and b the regularization is applied to the new variables: $\Re(\vec{\alpha})$. Second, the expansion (26) satisfies the stacking constraints. Other expansions often reduce to hamming distances in the binary domain, which do not satisfy the stacking constraints.

4. Experiments

4.1. Comparison to Regularization

The first synthetic problem is similar to the threshold max (TM) problem presented by Perkins et al. (2003). The problem has 100 dimensions, $\vec{x} \in \mathrm{R}^{100}$ and each dimension is uniformly distributed between -1.866 and 0.134. We use the first 10 dimensions of each sample to determine class labels according to the following rule:

$$y = \begin{cases} 1 \text{ if } \max(x_1, x_2, \ldots, x_{10}) \geq 0 \\ -1 \text{ otherwise} \end{cases} \tag{27}$$

The range of the uniform distribution is such that (27) will assign approximately half the samples to Class 1.

Table 1. Free parameters associated with the four methods.

Method	Sol^N	Regularization C	Margin γ
M1	LP	500	$[1, \ldots, 10]$
M2	QP	500	$[1, \ldots, 10]$
R1	LP	$[1.0, 0.5, 0.1, 0.05,$	0
R2	QP	$0.01, 5e-3, 1e-3$	0
		$5e-4, 1e-4, 5e-5]$	

Observe that after thresholding input vectors at 0, the problem is separable with a linear classifier and hence separable with a WOS classifier.

The second synthetic problem was used by Cannon et al. (2003). The problem has 50 dimensions and samples are drawn from Gaussian class conditional probabilities with equal covariance and class means:

$$\vec{\mu_1} = \vec{0} \text{ and } \vec{\mu_{-1}} = 0.276(\vec{1}) \tag{28}$$

The class marginal probabilities are $Pr(y = 1) = \frac{2}{3}$ and $Pr(y = -1) = \frac{1}{3}$ and the Bayes error for this problem is 0.15.

We investigate four methods to control complexity of the weighted order statistic classifier. These methods are summarized in Table 1. The first two methods use rank-order margin as the free parameter. The two rank-order methods, M1 and M2, differ in the type of regularization that is applied in (24). The R1 and R2 methods threshold the input data at zero, and then solve the binary input linear classifier design problem. Complexity is controlled by regularization and the free parameter is C from (24).

For each synthetic problem we will solve for both $\vec{w}$ and b in (25) and $\vec{\alpha}$ and b in (26). We have four problems in total, and four methods for each problem. Training and validation sets of size 30, 60, 120, 240, and 480 are investigated. The test set in all cases is 5000 samples. In each experiment, a training set is used to optimize a classifier at each value of the free parameter in Table 1. We choose the classifier that performs best on the validation set and calculate its error on the test set. Results were averaged over 10 trials and are summarized in Figure 4.

4.2. Comparison to Boosting

The second experiment is similar to one used by Mason et al. (2000) to investigate DOOM. We use four two-class data sets from the UCI Machine Learning Repository (Blake & Merz, 1998). Each problem is di-

Table 2. Testing, validation, and test set sizes.

Data Set	Training Samples	Validation and Testing Samples
Ionosphere	100	125
Sonar	58	75
Breast Cancer	182	250
Heart Disease	96	100

Table 3. Average test errors after model selection.

Data Set	Boosting	M1
Ionosphere	0.11	0.08
Sonar	0.32	0.32
Breast Cancer	0.35	0.35
Heart Disease	0.40	0.41

vided randomly into training, validation and test sets of size specified in Table 2.

We use data centered spheres (Cannon et al., 2003) to generate weak learners for the boosting procedure. A new pattern $\vec{x}$ is mapped to an N dimensional vector $k = [k_1, k_2, \ldots, k_N]$, $k_n \in \mathrm{R}$, based on training samples:

$$k_n = \|\vec{x} - \vec{x}(n)\| - t_n \tag{29}$$

for $n = 1, \ldots, N$ and $\| \bullet \|$ is the Euclidean distance. We use the adaboost m.1 algorithm to incrementally construct base hypotheses. The weak learner selects, through exhaustive search, k_n from k and the threshold t_n to minimize training error. For t_n we consider $N+1$ discrete values based on the training data (i.e. the average value of consecutive training points after projection). After 100 iterations we augment the model with mirrored (or negative) input samples. Weights are then reoptimized using the M1 method varying margin γ from 1 through to 50. We averaged over 10 trials and the training and test errors are shown in Figure 5.

During boosting we evaluate the ensemble at each iteration using the validation set. We choose the iteration with lowest error and report the test error for the 10 trials. We use the same approach to select the best value for rank-order margin and results are summarized in Table 3.

4.3. Discussion of Results

With only 10% of the features relevant, the TM-primal problem is solved best by the M1 and R1 methods with L_1 regularization. M2 had similar performance to R2 despite the fact that C was fixed at 500. This indicates the TM-primal problem is very sensitive to the choice of regularization. For the TM-expansion problem we observe a significant decrease in performance for all methods, which can be attributed to the large number of noisy or irrelevant features. For the Gaussian primal problem the R2 method had the best performance. In this experiment there was little difference between M1 and M2, and both had similar performance to R1. We see a similar result for the Gaussian-expansion problem, and in this case M1, M2 and R1 have similar performance to the Gaussian-primal R2 solution.

The γ-margin curves in Figure 5 are consistent with the hypothesis that rank-order margin is related to generalization error. The Breast Cancer result illustrates that rank-order margin sacrifices substantial training error as margin is increased to improve test error. Boosting and the M1 method had similar performance in Table 3. The largest improvement was made on the Ionosphere data set. However in Figure 5 we observe that boosting may not have converged and therefore with additional iterations the boosting result would be expected to improve.

5. Conclusions and Future Work

We have a presented a new concept of margin for stack filter derived function classes. Rank-order margin has three significant properties: (1) There are a finite set of rank-order margin loss functions that can be directly optimized. (2) Rank-order margin and L_1, L_2 regularization appear to be different, but related mechanisms for controlling WOS classifier complexity that obtain similar performance. (3) Rank-order margin requires real-valued inputs for training (21), even though the final binary domain classifier is applied to inputs threshold at zero. In contrast, regularization is applied directly to the binary domain classifier and does not consider the real-valued inputs. If the classification problem has only binary inputs, the rank-order margin mechanism cannot be applied.

Future research will determine if stack filter based classifiers and rank-order margin have a practical payoff. In the synthetic experiments we solved linear and quadratic programs that have similar complexity to those used to design linear classifiers. However, since WOS classifiers threshold inputs at zero before applying the linear classifier, approximation error can be high. Mapping to larger feature spaces can reduce approximation error, but this must be done efficiently (e.g. kernel methods) so that the optimization problem remains computationally reasonable. Constructive approaches like boosting are one way to search larger feature spaces efficiently. However, the features and thresholds found by boosting may not be ideal for the WOS classifier, and as observed in experiments, the performance gains may not be significant.

The approximation problem also manifests itself in nonlinear digital filtering and several researchers have suggested hyrbid linear / weighted order statistic filters to address the problem (Yin & Neuvo, 1994). Our future research will explore some of these ideas and aims to combine non-linear digital filtering and machine learning research efforts to benefit both fields of study.

Acknowledgements

This work was supported by Los Alamos National Lab's Deployable Adaptive Processing Systems and Real World Machine Learning projects.

References

Arce, G. R. (1998). A general weighted median filter structure admitting negative weights. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, *46*, 3195–3205.

Astola, J., & Kuosmanen, P. (1997). *Fundamentals of nonlinear digital filtering*. New York, NY: CRC Press LLC.

Blake, C. L., & Merz, C. J. (1998). Uci repository of machine learning databases. *University of California at Irvine, Dept. of Information and Computer Sciences.*

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Cannon, A., Howse, J., Hush, D., & Scovel, C. (2003). Simple classifiers. *Manuscript Submitted to: IEEE Transactions on Neural Networks.*

Chen, K. (1989). Bit-serial realizations of a class of nonlinear filters based on positive boolean functions. *IEEE Transactions on Circuits and Systems Recognition*, *36*, 785–794.

Fitch, J. P., Coyle, E. J., & Gallagher, N. C. (1984). Median filtering by threshold decomposition. *IEEE Transactions on Acoustics Speech and Signal Processing*, *32*, 1183–1188.

Grove, A. J., & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. *AAAI-98: Proceedings of the Fifteenth National Conference on Artifical Intelligence.*

Lin, J., & Coyle, E. J. (1990). Minimum mean absolute error estimation over the class of generalized stack filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *38*, 663–678.

Lin, L., III, G. B. A., & Coyle, E. J. (1994). Stack filter lattices. *Signal Processing*, *38*, 277–297.

Mason, L., Bartlett, P. L., & Baxter, J. (2000). Improved generalization through explicit optimization of margins. *Machine Learning*, *38*, 243–255.

Paredes, J. L., & Arce, G. R. (2001). Optimization of stack filters based on mirrored threshold decomposition. *IEEE Transactions on Signal Processing*, *49*, 1179–1188.

Perkins, S., Lacker, K., & Theiler, J. (2003). Grafting: Fast, incremental feature selection by gradient descent in function spac. *Journal of Machine Learning Research*, *3*, 1333–1356.

Preston, K. (1990). Detection of weak subpixel targets using mesh-connected cellular automata. *IEEE Transactions on Aerospace and Electronic Systems*, *26*, 548–558.

Ritter, G. X., & Sussner, P. (1996). An introduction to morphological neural networks. *13th International Conference on Pattern Recognition*, *4*, 709–717.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, *26*, 1651–1686.

Tumer, K., & Ghosh, J. (1999). Linear and order statistics combiners for pattern classification. *Combining Artificial Neural Nets.*, 127–162.

Wendt, P. D., Coyle, E. J., & Gallagher, N. C. (1986). Stack filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *34*, 898–911.

Yang, P., & Maragos, P. (1995). Min-max classifiers: Learnability, design and application. *Pattern Recognition*, *28*, 879–899.

Yin, L., & Neuvo, Y. (1994). Fast adaptation and performance characteristics of fir-wos hybrid filters. *IEEE Transactions on Signal Processing*, *42*, 1610–1628.

Yli-Harja, O., Astola, J., & Neuvo, Y. (1991). Analysis of the properties of median and weighted median filters using threshold logic and stack filter representation. *IEEE Transactions on Signal Processing*, *39*, 395–410.

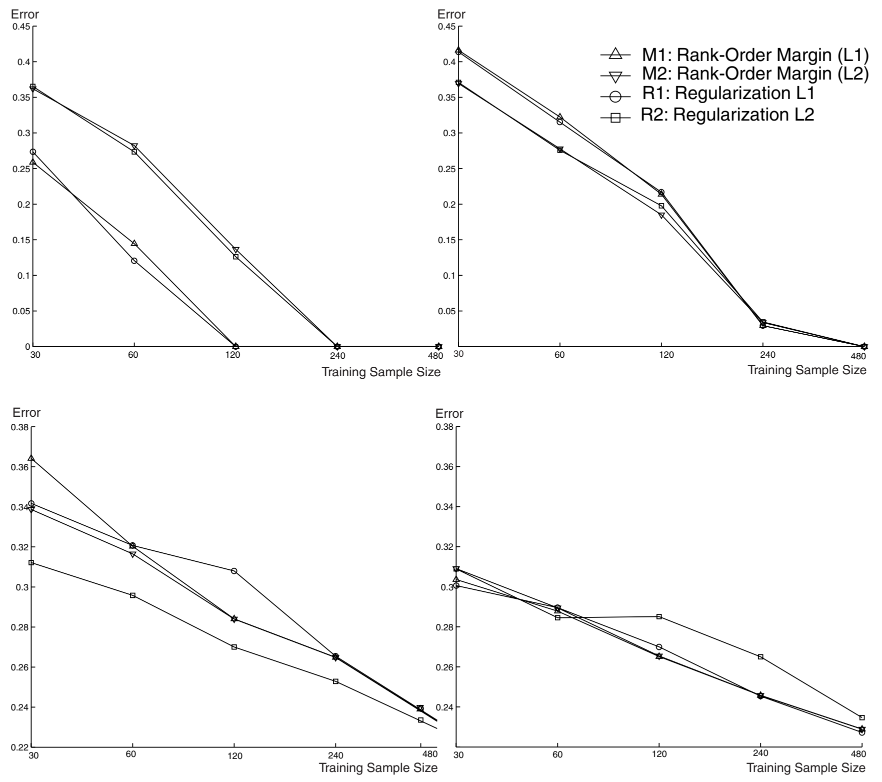

Figure 4. (top left) TM-primal, (top right) TM-expansion, (bottom-left) Gaussian-primal and (bottom right) Gaussian-expansion.

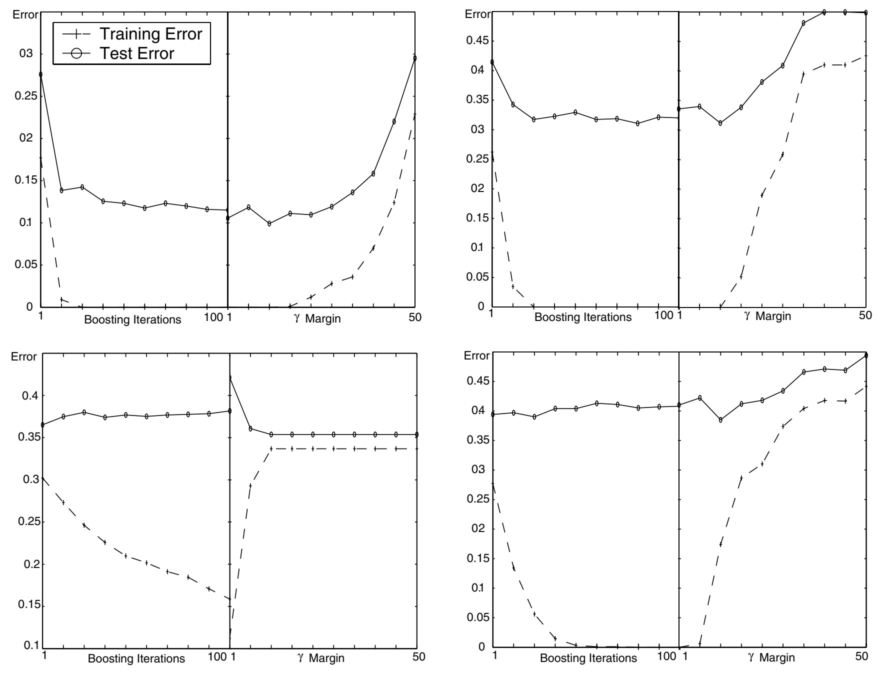

Figure 5. (top left) Ionosphere, (top right) Sonar, (bottom left) Breast Cancer and (bottom right) Heart Disease.