Polyhedral Classifier for Target Detection
A Case Study: Colorectal Cancer

M. Murat Dundar MURAT.DUNDAR@SIEMENS.COM
Matthias Wolf MWOLF@SIEMENS.COM
Sarang Lakare SARANG.LAKARE@SIEMENS.COM
Marcos Salganicoff MARCOS.SALGANICOFF@SIEMENS.COM
Vikas C. Raykar VIKAS.RAYKAR@SIEMENS.COM
CAD & Knowledge Solutions, Siemens Medical Solutions Inc., Malvern, PA 19355 USA

Abstract

In this study we introduce a novel algorithm for learning a polyhedron to describe the target class. The proposed approach takes advantage of the limited subclass information made available for the negative samples and jointly optimizes multiple hyperplane classifiers each of which is designed to classify positive samples from a subclass of the negative samples. The flat faces of the polyhedron provides robustness whereas multiple faces contributes to the flexibility required to deal with complex datasets. Apart from improving the prediction accuracy of the system, the proposed polyhedral classifier also provides run-time speedups as a by-product when executed in a cascaded framework in real-time. We evaluate the performance of the proposed technique on a real-world Colon dataset both in terms of prediction accuracy and online execution speed.

1. Problem Specification

In target detection the objective is to determine whether or not a given example is from a target class. Obtaining ground truth for the target class usually involves a tedious process of manual labeling. If samples belonging to the target class are labeled as positive, then negative class covers everything else. Due to the nature of the problem and the labeling process, the number of samples representing the target class is usually scarce whereas abundant data is potentially available to represent the negative class. In other words the data is highly unbalanced between classes favoring the negative class.

In this process the actual labels of the counter-examples are ignored and the negative class is formed by pooling samples of potentially different characteristics together within a single class. In other words samples of the negative class do not cluster well since they can belong to different subclasses.

One promising approach that has been heavily explored in this domain is the one-class classifiers. One-class classification simply omits the negative class (if it exists) and aims to learn a model with the positive examples only. Several techniques have been proposed in this direction. Support vector domain description technique aims to fit a tight hyper-sphere in the feature space to include most of the positive training samples and reject outliers (Tax & Duin, 1999). In this approach the nonlinearity of the data can be addressed implicitly through the kernel evaluation of the technique. One-class SVM generates an artificial point through kernel transformation for representing the negative class and then using relaxation parameters it aims to separate the image of the one-class from the origin (Scholkopf et al., 1999). Compression Neural Network constructs a three-layer feed-forward neural network and trains this network with a standard back-propogation algorithm to learn the identity function on the positive examples (Manevitz & Yousef, 2001).

Discriminative techiques such as Support Vector Machines (Vapnik, 1995), Kernel Fisher Discriminant (Mika et al., 2000), Relevance Vector Machines (Tipping, 2000) to name few are also used in this domain. These techniques deal with the unbalanced nature of the data by assigning different cost factors to the negative and positive samples in the objective function. The kernel evaluation of these techniques yields nonlinear decision boundaries suitable for classifying multi-mode data from the target class.

In this study we aim to learn a polyhedron in the feature

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

space to describe the positive training samples. Polyhedral decision boundaries such as boundaries that are drawn parallel to the axes of the feature space as in decision trees or skewed decision boundaries (Murth et al., 1994) have existed for quite some time. Our approach is similar in some sense to the Support vector domain description technique but there are two major differences. First instead of a hypersphere, a polyhedron is used to fit positive training samples. Second, positive and negative samples are used together in this process. The target polyhedron is learned through joint optimization of multiple hyperplane classifiers, each of which is designed to classify positive samples from a subgroup of negative samples. The number of such hyperplane classifiers is equivalent to the number of subclasses identified in the negative class. The proposed technique requires labeling of a small portion of the negative samples to collect training data for the subclasses that exist in the negative class.

Our approach does not intend to precisely identify each and every subclass in the dataset. By manual labeling we aim to identify major subclasses. Subclasses with similar characteristics or with only few labeled samples can be grouped together. During annotation one may also encounter positive look alikes, i.e. samples do not appear as negative but not yet confirmed as positive. A new subclass can be introduced for these samples.

In Figure 1 the proposed algorithm is demonstrated with a toy example. Positive samples are depicted by the dark circles in the middle, whereas negative samples are depicted with the numbers with each number corresponding to a different subclass. All eight classifiers are optimized simultaneously and polygon shown with dark lines is obtained as a decision boundary that classifies positive samples from the negative ones.

Kernel-based classifiers have the capacity to learn higly nonlinear decision boundaries allowing great flexibility. However it is well-known that in real-world applications where feature noise and redundancy is a problem, too much capacity usually hurts the generalizability of a classifier by enabling the classifier to easily overfit the training data. The proposed approach is capable of addressing nonlinearities by fitting the positive class through a series of linear hyperplanes, all of which are optimized jointly to form a polyhedral decision boundary. The flat faces provides robustness whereas multiple faces contributes to the flexibility.

The problem described above is commonly encountered in areas like content-base image retrieval (Chen et al., 2001), document classification (Manevitz & Yousef, 2001) and speech recognition (Brew et al., 2007). A similar scheme is also observed in Computer Aided Detection (CAD). In this study we explore the proposed idea for a CAD application, namely Colon CAD.

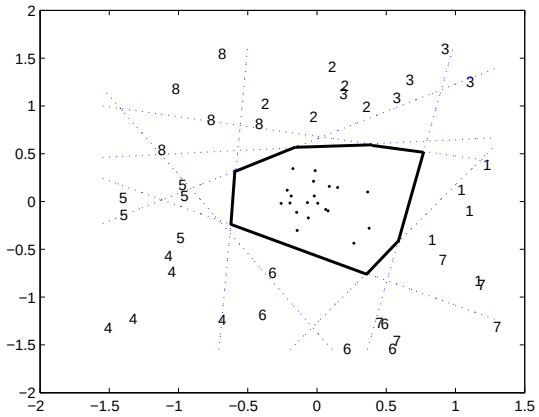

Figure 1. A Toy example demonstrating the proposed algorithm. Dark circles depicting positive samples, numbers representing negative samples. The decision boundary is shown with the solid lines.

2. Hyperplane Classifiers

We are given a training dataset $\{(x_i, y_i)\}_{i=1}^{\ell}$, where $x_i \in \Re^d$ are input variables and $y_i \in \{-1, 1\}$ are class labels. We consider a class of models of the form $f(x) = \alpha^T x$, with the sign of $f(x)$ predicting the label associated with the point x. An hyperplane classifier with hinge loss can be designed by minimizing the following cost function.

$$\mathcal{J}(\alpha) = \Phi(\alpha) + \sum_{i=1}^{\ell} w_i \left(1 - \alpha^T y_i x_i\right)_+ \qquad (1)$$

where the function $\Phi : \Re^{(d)} \Rightarrow \Re$ is a regularization function or regularizer on the hyperplane coefficients and $(k)_+ = max(0, k)$ represents the hinge loss, and $\{w_i : w_i \geq 0, \forall i\}$ is the weight preassigned to the loss associated with x_i. For balanced data usually $w_i = w$, but for unbalanced data it is a common practice to weight positive and negative classes differently, i.e. $\{w_i = w_+,\ \forall i \in C^+\}$ and $\{w_i = w_-,\ \forall i \in C^-\}$ where C^+ and C^- are the corresponding sets of indices for the positive and negative classes respectively.

The function $\left(1 - \alpha^T y_i x_i\right)_+$ is a convex function. The weighted sum of convex functions is also convex. Therefore for a convex function $\Phi(\alpha)$ (1) is also convex. The problem in (1) can be formulated as a mathematical pro-

gramming problem as follows:

$$\begin{array}{rcl} \min\limits_{(\alpha,\xi)\in R^{d+\ell}} \quad \Phi(\alpha) & + & \sum_{i=1}^{\ell} w_i\xi_i \\ \text{s.t.} \quad \xi_i & \geq & 1-\alpha^T y_i x_i \\ \xi_i & \geq & 0,\ \forall i \end{array} \tag{2}$$

For $\Phi(\alpha) = \|\alpha\|_2^2$, where $\|.\|_2$ is the 2-norm, (2) results in the conventional Quadratic-Programming-SVM, and for $\Phi(\alpha) = |\alpha|$, where $|.|$ is the 1-norm it yields the sparse Linear-Programming-SVM.

3. Polyhedral Decision Boundaries

3.1. Training a Classifier with an AND Structure

We aim to optimize the following cost function

$$\begin{aligned} \mathcal{J}(\alpha_1,\dots\alpha_K) &= \sum_{k=1}^{K}\Phi_k(\alpha_k) \\ &+ \nu_1\sum_{k=1}^{K}\sum_{i\in C_k^-}(e_{ik})_+ \\ &+ \nu_2\sum_{i\in C^+}\max(0,e_{i1},\dots,e_{iK}) \end{aligned} \tag{3}$$

where $e_{ik} = 1+\alpha_k^T x_{ik}$ and $(e_{ik})_+$ defines the hinge loss of the i-th training example $\{(x_{ik}, y_{ik})\}$ in subclass-k induced by classifier k. C_k^- is the set of indices of the negative samples in subclass-k. Note that classifier k is designed to classify positive examples from the negative examples in the subclass-k. The first term in (3) is a summation of the regularizers for each of the classifiers in the cascade and the second and third terms accounts for the losses induced by the negative and positive samples respectively. Unlike (1) the loss function here is different for the positive samples. The loss induced by a positive sample i, $i \in C^+$ is zero only if $\forall k : 1-\alpha_k^T x_i \leq 0$, which corresponds to the "AND" operation. The problem (3) can be formulated as follows

$$\begin{array}{rcl} \min\limits_{(\alpha,\xi)\in R^{Kd+\ell}} \quad \sum_{k=1}^{K}\Phi_k(\alpha_k) & + & \nu_1\sum_{k=1}^{K}\sum_{i\in C_k^-}\xi_{ik} \\ & + & \nu_2\sum_{i\in C^+}\xi_i \\ \\ \text{s.t.} \quad \xi_{ik} & \geq & 1+\alpha_k^T x_{ik} \\ \xi_{ik} & \geq & 0 \\ \xi_i & \geq & 1-\alpha_k^T x_i \\ \xi_i & \geq & 0 \end{array} \tag{4}$$

where the first two constraints are imposed for $\forall i \in C_k^-$, $k = 1,...,K$ and the last two constraints are imposed for $\forall i \in C^+$, $k = 1,...,K$. Note that for a convex function $\Phi(\alpha)$ the problem in (4) is convex. In a nutshell we designed K classifiers, one for each of the binary classification problems, i.e. positive class vs subclass-k of the negative class. Then we construct a learning algorithm to jointly optimize these classifiers such that the cost induced by a positive sample is zero if and only if all of the K classifiers classifies this sample correctly, i.e. $\forall k : 1-\alpha_k^T x_i \leq 0$. Since each negative sample is only used once for training the classifier k, the cost induced for a negative sample is zero as long as it is classified correctly by the corresponding classifier k, i.e. $1+\alpha_k^T x_{ik} \leq 0$. Each classifier can use an arbitrary subset of the original feature set. This provides run time advantages in real-time when the classification architecture is implemented in a cascaded framework. This will be explained later in the paper. For now to keep the notation clean and tractable we assumed each classifier uses the entire feature set in the formulation (4) above.

3.2. Training a Classifier with an AND-OR Structure

The AND algorithm is developed with the assumption that the negative class is fully labeled. That is to say, the subclass membership of each of the negative sample is known apriori. For most real world applications this is not a very realistic scenario as it is almost impractical to label all of the negative samples due to the time limitations. However one can label a small subset of the negative class to discover different type of subclasses as well as pool the training data for each subgroup. To accommodate for the unlabeled samples we modify the equation (3) for the unlabeled negative samples as follows.

$$\begin{aligned} \mathcal{J}(\alpha_1,\dots\alpha_K) &= \sum_{k=1}^{K}\Phi_k(\alpha_k) \\ &+ \nu_1\sum_{k=1}^{K}\sum_{i\in C_k^-}(e_{ik})_+ \\ &+ \nu_1\sum_{i\in \acute{C}^-}\prod_{k=1}^{K}(e_{ik})_+ \\ &+ \nu_2\sum_{i\in C^+}\max(0,e_{i1},\dots,e_{iK}) \end{aligned}$$

where $\acute{C}^-$ is the set of indices of the unlabeled negative samples. The first term in (5) requires a labeled sample from a subclass-k to be correctly classified by the classifier k. In other words if a sample is known to be a member of subclass-k, ideally it should be classified as negative by the corresponding classifier k. On the other hand the second term requires an unlabeled negative sample to be correctly classified by any of the classifiers. As long as an unlabeled sample is classified as negative it does not matter which classifier does it, i.e. $\exists k : 1-\alpha_k^T x_{ik} \leq 0$ which corresponds to a "OR" operation. The third term requires a positive sample is classified as positive by all of the K classifiers, i.e. $\forall k : 1-\alpha_k^T x_i \leq 0$, which corresponds to the "AND" operation.

Due to the product operation in the objective function for unlabeled samples, unlike equation (3), equation (5) can not be cast as a convex programming problem. In the next section we propose an efficient alternating optimization algorithm to solve this problem.

4. Cyclic Optimization of AND-OR Algorithm

We develop an iterative algorithm which, at each iteration, carries out K steps, each aiming to optimize one classifier at a time. This type of algorithms is usually called alternating or cyclic optimization approaches. At any iteration, we fix all of the classifiers but the classifier k. The fixed terms have no effect on the optimization of the problem once they are fixed. Hence solving (5) is equivalent to solving the following problem by dropping the fixed terms in (5):

$$\begin{aligned}\mathcal{J}(\alpha_k) &= \Phi_k(\alpha_k) \\ &+ \nu_1 \sum_{i\in C_k^-} (e_{ik})_+ \\ &+ \nu_1 \sum_{i\in \acute{C}^-} w_i\ (e_{ik})_+ \\ &+ \nu_2 \sum_{i\in C^+} \max(0, e_{i1}, \ldots, e_{ik}, \ldots, e_{iK})\end{aligned}$$

where $w_i = \prod_{k=1,k\neq k}^{K} (e_{ik})_+$. This can be cast into a constrained problem as follows

$$\begin{aligned}\min_{(\alpha_k,\xi)\in R^{d+\ell_k+\ell_+}} \quad & \Phi_k(\alpha_k) + \nu_1 \textstyle\sum_{i\in C_k^-} \xi_i \\ & + \nu_1 \textstyle\sum_{i\in \acute{C}^-} w_i\xi_i \\ & + \nu_2 \textstyle\sum_{i\in C^+} \xi_i \\ \text{s.t.} \quad & \xi_i \geq e_{ik},\ \forall i \\ & \xi_i \geq 0,\ \forall i \in \acute{C}^- \cup C_k^- \\ & \xi_i \geq \gamma_i,\ \forall i \in C^+\end{aligned} \tag{5}$$

where $\gamma_i = max\left(0, e_{i1}, \ldots, e_{i(k-1)}, e_{i(k+1)}, \ldots, e_{iK}\right)$ and ℓ_k is the number of samples in subclass-k. The subproblem in (5) is a convex problem and differs from the problem in (2) by two small changes. First the weight assigned to the loss induced by the negative samples is now adjusted by the term $w_i = \prod_{k=1,k\neq k}^{K} (e_{ik})_+$. This term multiplies out to zero for negative samples correctly classified by one of the other classifiers. For these samples $e_{ik} < 0$ and $\xi_i = 0$ making the constraints on ξ_i in (5) redundant. As a result there is no need to include these samples when training for the *classifier-k*, which yields significant computational advantages. Second the lower bound for ξ is now $max\left(0, e_{i1}, \ldots, e_{i(k-1)}, e_{i(k+1)}, \ldots, e_{iK}\right)$. This implies that if any of the classifiers in the cascade misclassifies x_{ik} the lower bound on ξ is no longer zero relaxing the constraint on x_{ik}.

4.1. An Algorithm for AND-OR Learning

(0) Initialize e_{ik} in (5) such that all candidates are classified as positive, i.e. 100% sensitivity, 0% specificity. Set counter $c = 1$.

(i) Fix all the classifiers in the cascade except classifier k and solve (5) for α_k^c using the training dataset $\left\{\left(x_i^k, y_i\right)\right\}_{i=1}^{\ell}$. Repeat this for all $k = 1, \ldots, K$.

(ii) Compute $J^c(\alpha_1, \ldots, \alpha_K)$ by replacing α_k^{c-1} by α_k^c in (5), for all $k = 1, \ldots, K$.

(iii) Stop if $J^c - J^{c-1}$ is less than some desired tolerance. Else replace α_k^{c-1} by α_k^c for all $k = 1, \ldots, K$, c by $c + 1$ and go to *step i*.

The initial objective function in (5) is neither convex nor twice differentiable due to the product of the hinge loss term. Therefore the convergence theorem introduced in (Bezdek & Hathaway, 2003) for cyclic optimization does not hold here. On the other hand the subproblem in (5) is convex and hence at each iteration $J^c <= J^{c-1}$ holds and also (5) is bounded below. These guarantee convergence of the algorithm from any initial point to the set of suboptimal solutions. The solution is suboptimal if the objective function J can not be further improved following any directions. For a more detailed discussion on this topic please see (Dundar & Bi, 2007).

An unseen sample x can be classified as positive if $max(1 - \alpha_1^T x, \ldots, 1 - \alpha_K^T x) \leq \tau$ and as negative if vice versa for a threshold τ. The receiver operating characteristics (ROC) curve can be plotted by varying the value of τ.

5. Cascade Design for Run-Time Speedups

In Figure 2 a cascade classification scheme is shown. The key insight here is to reduce the computation time and speed-up online learning. This is achieved by designing simpler classifiers in the earlier stages of the cascade to reject as many negative candidates as possible before calling upon classifiers with more features to further reduce the false positive rate. A positive result from the first classifier activates the second classifier and a positive result from the second classifier activates the third classifier, and so on (Viola & Jones., 2004). A negative outcome for a candidate at any stage in the cascade leads to an immediate rejection of that candidate. Under this scenario $T_{k-1} = T_k \cup F_k$ and $T_0 = T_K \cup \bigcup_1^K F_k$ where T_k and F_k are the sets of candidates labeled as positive and negative respectively by classifier k.

The proposed algorithm learns a polyhedron through jointly optimizing a series of sparse linear classifiers. Since

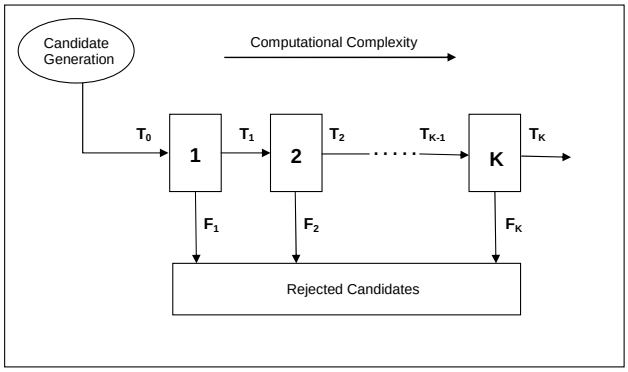

Figure 2. A general cascade framework used for online classification

the order these classifiers are executed in real-time does not matter in terms of the overall prediction accuracy of the system, we can arrange the execution sequence of these classifiers in a way to optimize the run-time. Let Γ_k be the set of indices of nonzero coefficients for *classifier-k*, t_i be the time required to compute *feature i* for a given candidate, $i = 1, \ldots, d$ and n_{k-1} is the number of samples in set T_{k-1}, then the total time required for the online execution of the algorithm is

$$\mathcal{T} = \sum_{k=1}^{K} n_{k-1} \sum_{i \in \left(\Gamma_{k-1} \setminus \bigcup_{i=0}^{k-2} \Gamma_i\right)} t_i \qquad (6)$$

The sets Γ_k are learned during offline training of the polyhedral classifier and are fixed for online execution. However the sets T_k is a function of the *classifier 1* through *classifier k-1*. Therefore this is a combinatorial optimization problem with $K!$ different outcomes, where $K!$ is the factorial of K. For small K one can try the exhaustive number of orderings between classifier to find the optimum sequence. However when K is large we can start with the *most sparse* classifier, i.e. the linear classifier with the least number of nonzero coefficients and choose the next classifier as the one that will require computing least number of additional features.

6. Computer Aided Detection

The goal of a Computer Aided Detection (CAD) system is to detect potentially malignant tumors and lesions in medical images (CT scans, X-ray, MRI etc). In an almost universal paradigm for CAD algorithms, this problem is addressed by a 3 stage system: A typical CAD system consists of a candidate generation phase, a feature extraction module and a classifier. The task of the candidate generation module is to create a list of potential polyps with a high sensitivity but low specificity. Features are then extracted for each candidate and eventually passed to a classifier where each candidate is labeled as normal or diseased.

In order to train a CAD system, a set of medical images (eg CT scans, MRI, X-ray etc) is collected from archives of community hospitals that routinely screen patients, *e.g.* for colon cancer. These medical images are then read by expert radiologists; the regions that they consider unhealthy are marked as ground-truth in the images. After the data collection stage, a CAD algorithm is designed to learn to diagnose images based on the expert opinions of the radiologists on the database of training images. First, domain knowledge engineering is employed to (a) identify all potentially suspicious regions in a candidate generation stage, and (b) to describe each such region quantitatively using a set of medically relevant features based on for example, texture, shape, intensity and contrast. Then, a classifier is trained using the features computed for each candidate in the training data and the corresponding ground truth.

When training a classifier for a CAD system for detection of colonic polyps, the only information that is usually available is the location of polyps, since radiologists only mark unhealthy regions when they are reading cases. This, of course, is very important for training a CAD system. But for all other structures that are picked up during candidate generation phase that are not pointing to a known lesion there is no other information available and they all have to be treated equally as negative examples. This introduces two complications. First all the negative candidates are clustered in one group although variation in size and shape among them is very big and valuable information about those candidates, e.g. type category, is not used. Second, radiologists only mark lesions of clinical importance, i.e. polyps greater than 6mm in colon. It is also possible that some lesions are overlooked during clinical evaluation. So potentially there are unidentified lesions in our dataset with no matching ground truth. If the candidate generation algorithm generates candidates for these lesions then these candidates are also marked as negative together with all the other candidates with no corresponding radiologist marks. In other words negative class may also contain unidentified samples of the target class.

In the rest of this section we will discuss some motivation for the proposed algorithm within the scope of a CAD system designed to detect colorectal cancer. Colorectal cancer is the second leading cause of cancer-related death in the western world (Jemal et al., 2004). Early detection of polyps through colorectal screening can help to prevent colon cancer by removing the polyps before they can turn malignant.

Typical examples of different polyp morphologies are given in Figure 3.

The commonly encountered false positive types in colon are fold, stool, tagged stool, meniscus, illeocecal valve, rectum etc. Some of these are shown in Figure 4. Ideally we can label all of the negative candidates in the training data and use the proposed AND algorithm in (4) to jointly train classifiers one for each of the subclasses of negative samples. However an exhaustive annotation of all negative examples is not feasible. Therefore we first select a very small subset of the negative candidates and annotate them manually through visual inspection. Then this set together with the positive samples and the remaining negative samples, i.e. unlabeled samples is used in the proposed $AND-OR$ framework to train the classifiers.

7. Experimental Results

We validate the proposed polyhedral classifier *(*polyhedral) with respect to its generalization performance and run-time efficiency. We compared our algorithm to a Support Vector Domain Description technique *(*svdd) (Tax & Duin, 1999), nonlinear SVM with Radial Basis Function *(*rbf), and one-norm SVM *(*sparse). To achieve sparseness we set the $\Phi_k(\alpha_k) = |\alpha_k|$ for the polyhedral classifier.

7.1. Data and Experimental Settings

The database of high-resolution CT images used in this study were obtained from two different sites across US. The 370 patients were randomly partitioned into two groups: training (n=167) and test (n=199). The test group was sequestered and only used to evaluate the performance of the final system.

Training Data Patient and Polyp Info: There were 167 patients with 316 volumes. The candidate generation (CG) algorithm identifies a total of 226 polyps at the volume level across all sizes while generating a total of 64890 candidates or an average of 205 false positives per volume. *Testing Data Patient and Polyp Info:* There were 199 patients with 385 volumes. The candidate generation (CG) algorithm identifies a total of 245 polyps at the volume level across all sizes while generating an average of 75946 samples or 194 false positives per volume (fp/vol).

A total of 98 features are extracted to capture shape and intensity characteristics of each candidate. The proposed algorithm requires a small set of false positives annotated. Rather than labeling false positives randomly across a dataset with 64890 samples we used the output of the most recent prototype classifier for labeling. This classifier is trained using a naive SVM and optimized for the 0-5 fp/vol range. This way we only focus on the most challenging false positives. This classifier marks a total of 1432 candidates as positive. Out of these candidates 1249 are false positives. A small subset of the volumes from this set is chosen for labeling and a total of 177 false positives (out of 1249) are annotated and ten different subcategories are identified.

7.2. Performance Evaluation

The classifiers are trained with the combination of 1249 false positives generated by the prototype classifier and all the polyps the candidade generation detects in the training data. A total of 1560 candidates are used for training. Classifiers are evaluated on the 1920 candidates the prototype classifier marked as positive in the testing data. The corresponding Receiver Operating Characteristics (ROC) curves for each algorithm is plotted in Figure 5.

The classifier parameters are estimated using a 10-fold patient cross validation from a set of discrete values using the training data. These are namely the width of the kernel (γ=[0.01 0.03 0.05 0.1 0.3 0.5 1 5]) for *rbf*, *svdd*, the cost factor (c=[5 10 15 20 25 50 75 100]) for *rbf*, *polyhedral*, *sparse* and the ν=[0.001 0.005 0.01 0.05 0.1 0.2] parameter for *svdd*. The desired tolerance value for Algorithm 4.1 is set to 0.001. The algorithm converged in less than 10 iterations.

As shown in Figure 5 the ROC curve corresponding to the proposed *polyhedral* classifier is consistently dominating all the other curves. The curve associated with the *sparse SVM* is almost linear implying a random behavior. This is not surprising to a greater extent as both the training and testing data sets used in this experiment are derived from the initial datasets via a linear SVM classifier. In other words the datasets are composed of samples marked as positive by the linear SVM, a significant portion of which are false positives. The *one-class SVM* is only slightly better than the *sparse SVM*. Even though the *rbf SVM* is the best of the three competitor algorithms, the difference in sensitivity between the *rbf SVM* and the proposed *polyhedral* classifier can be as high as 5% (10 polyps) in favor of the proposed algorithm.

7.3. Run-time Evaluation

As stated earlier in the paper, run-time speedups can be achieved as a by-product of the proposed algorithm when the real-time classification is implemented in a cascaded framework as in Figure 2. For a more detailed discussion of cascade classifiers in the CAD domain we refer the interested readers to these recent works (Dundar & Bi, 2007), (Bi et al., 2006).

To avoid any delays in the workflow of a physician the CAD results should be ready by the time physician completes his own review of the case. Therefore there is a run-time requirement a CAD system needs to satisfy. Among the stages involved during online processing of a volume,

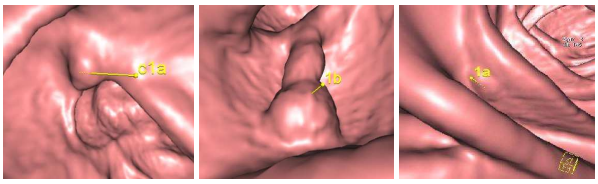

Figure 3. Polyp morphologies (from left to right): Sessile, pedunculated, and flat polyp

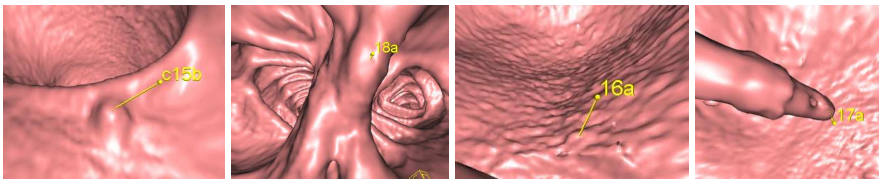

Figure 4. Negative examples (from left to right): stool, fold, noisy data and rectal tube

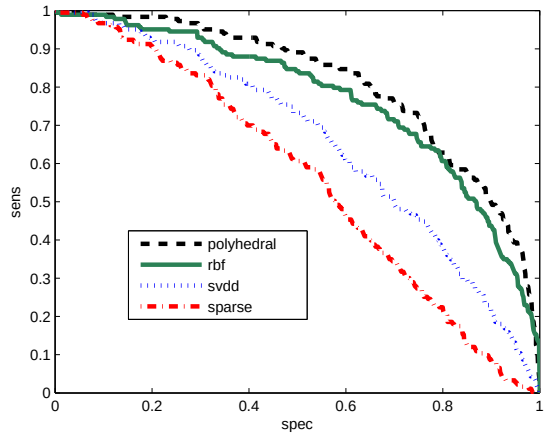

Figure 5. ROC curves obtained by the four classifiers on the test data.

feature computation is by far the most computational stage of online processing. A cascaded framework for executing the classifier in the order of increasing feature complexity may bring significant computational advantages in this case. In this framework the cascade is designed so as to execute classifiers with less number of features earlier in the sequence. This way the additional features required for the succeeding classifiers will only be computed on the remaining candidates, i.e. candidates marked as positive by all of the previous classifiers.

In this section we evaluate the speedups achieved by the proposed classifier. We set the operating point at 60% specificity, around 2.2fp/vol. At this specificity the proposed polyhedral classifier yields 85% sensitivity (see Figure 5). Assuming each feature takes on the average t seconds per candidate to compute we came up with the table in 1. This table shows the aggregate number of features used, feature computation time and number of candidates rejected at each stage in the sequence.

Feature computation for the proposed approach on average takes 452t secs per volume. On the other hand for *svdd* and *rbf*, which require computation of all the features at once, this stage takes 595t secs. This represents a roughly 25% improvement in run-time execution speed of the system. For the one-norm SVM *sparse*, this time is 437t secs. However the corresponding sensitivity at this operating point for one-norm SVM is around 40% vs 85% for the proposed technique.

8. Conclusions

In this study we have presented a methodology to take advantage of the subclass information information available in the negative class to achieve a more robust description of the target class. The subclass information which is neglected in conventional binary classifiers provides a better insight of the dataset and when incorporated into the learning mechanism acts as an implicit regularizer on the classifier coefficients. We believe this is an important contribution for applications where feature noise is prevalent. Highly nonlinear kernel classifiers provides flexibility for modeling complex data but they tend to overfit when there are too many redundant and irrelevant features in the data. Linear classifiers on the other hand do not have enough capacity to model complex data but they are more robust when there is noise. The polyhedral classifier is proposed as a midway solution. The linear faces of the polyhedron achieves robustness whereas multiple faces provides flexibility.

The order in which the classifiers are executed during online execution does not matter. Even though finding the globally optimum sequence is an open research problem for a large number of subclasses, the ordering of the clas-

sequence order	1	2	3	4	5	6	7	8	9
aggregate number of features	48	67	73	75	76	78	81	84	87
aggregate number of rejected candidates (avg. per volume)	1.08	2.37	2.57	2.82	2.90	2.93	3.06	3.07	3.40
aggregate feature computation time in t (avg. per volume)	291	386	408	414	418	424	434	443	452

Table 1. Run-time Results obtained for the Polyhedral classifier. The classifiers are executed in the order of increasing number of features required by each classifier.

sifiers can be arranged in a cascaded manner to reduce the total run-time of the system.

References

Bezdek, J., & Hathaway, R. (2003). Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, *11*, 351–368.

Bi, J., Periaswamy, S., Okada, K., Kubota, T., Fung, G., Salganicoff, M., & Rao, R. B. (2006). Computer aided detection via asymmetric cascade of sparse hyperplane classifiers. *Proceedings of the Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 837–844). Philadelphia, PA.

Brew, A., Grimaldi, M., & Cunningham, P. (2007). *An evaluation of one-class classification techniques for speaker verification* (Technical Report UCD-CSI-2007-8). University College Dublin.

Chen, Y., Zhou, X. S., & Huang, T. S. (2001). One-class svm for learning in image retrieval. *ICIP (1)* (pp. 34–37).

Dundar, M., & Bi, J. (2007). Joint optimization of cascaded classifiers for computer aided detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–8).

Jemal, D., Tiwari, R., Murray, T., Ghafoor, A., Saumuels, A., Ward, E., Feuer, E., & Thun, M. (2004). Cancer statistics.

Manevitz, L. M., & Yousef, M. (2001). One-class svms for document classification. *Journal of Machine Learning Research*, *2*, 139–154.

Mika, S., Rätsch, G., & Müller, K.-R. (2000). A mathematical programming approach to the kernel fisher algorithm. *NIPS* (pp. 591–597).

Murth, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of obligue decision trees. *Journal of Artificial Intelligence Research*, *2*, 1–33.

Scholkopf, B., Platt, O., Shawe-Taylor, J., Smola, A., & Williamson, R. (1999). Estimating the support of a high-dimensional distribution.

Tax, D. M. J., & Duin, R. P. W. (1999). Support vector domain description. *Pattern Recognition Letters*, *20*, 1191–1199.

Tipping, M. E. (2000). The relevance vector machine. In S. Solla, T. Leen and K.-R. Muller (Eds.), *Advances in neural information processing systems 12*, 652–658. Cambridge, MA: MIT Press.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.

Viola, P., & Jones., M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*.