Joint Optimization of Cascaded Classifiers for Computer Aided Detection

M. Murat Dundar
Knowledge Solutions and CAD
Siemens Medical Solutions Inc. USA
51 Valley Stream Parkway
Malvern, PA 19355
murat.dundar@siemens.com

Jinbo Bi
Knowledge Solutions and CAD
Siemens Medical Solutions Inc. USA
51 Valley Stream Parkway
Malvern, PA 19355
jinbo.bi@siemens.com

Abstract

The existing methods for offline training of cascade classifiers take a greedy search to optimize individual classifiers in the cascade, leading inefficient overall performance. We propose a new design of the cascaded classifier where all classifiers are optimized for the final objective function. The key contribution of this paper is the AND-OR framework for learning the classifiers in the cascade. In earlier work each classifier is trained independently using the examples labeled as positive by the previous classifiers in the cascade, and optimized to have the best performance for that specific local stage. The proposed approach takes into account the fact that an example is classified as positive by the cascade if it is labeled as positive by all the stages and it is classified as negative if it is rejected at any stage in the cascade. An offline training scheme is introduced based on the joint optimization of the classifiers in the cascade to minimize an overall objective function.

We apply the proposed approach to the problem of automatically detecting polyps from multi-slice CT images. Our approach significantly speeds up the execution of the Computer Aided Detection (CAD) system while yielding comparable performance with the current state-of-the-art, and also demonstrates favorable results over Cascade AdaBoost both in terms of performance and online execution speed.

1. Problem Specification

Cascade classifiers have long been used in machine learning for real-time object detection from images.

The typical steps involved in object detection are:

1. *Identify candidate structures in the image*: In 2D object detection this is achieved by taking sub-windows of the image at different scales. Each sub-window is a candidate. In 3D lesion detection from medical images, particularly image volumes generated by high-resolution computed tomography (CT), a more sophisticated candidate generation algorithm based on the image/shape characteristics of the lesion is required.

2. *Extract features for each candidate*: A number of image features are usually calculated to describe the target structure.

3. *Classify candidates as positive or negative*: A previously-trained classifier is used to label each candidate.

High sensitivity (ideally close to 100%) is essential, because any target structure missed at this stage can never be found by the system, which otherwise may be detected later in the classification stage by exploring effective features. Hence, a lot of false positives are generated in step 1 above (less than 1% of the candidates are positive), which makes the classification problem highly unbalanced. In Figure 1 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 yet highly sensitive classifiers in the earlier stages of the cascade to reject as many negative candidates as possible before calling upon classifiers with more complex 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 [10]. 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.

2. Brief Overview of Cascaded Methods

Previous cascade classification approaches are mostly based on AdaBoost [4, 9, 6]. Cascade AdaBoost serves as a great tool for building real-time robust applications [11, 10], especially for object detection systems.

However, cascade AdaBoost works with two implicit assumptions: 1. a significant amount of representative data is available for training the cascade classifier; 2. all features can be equally evaluated with a relatively low computational cost. These assumptions, unfortunately, often do not hold in practice. Available data can be noisy and hardly represent all aspects of the target characteristics. One of the major concerns about cascade classification approaches is if a classifier within the cascade does not generalize well and

1-4244-1180-7/07/$25.00 ©2007 IEEE

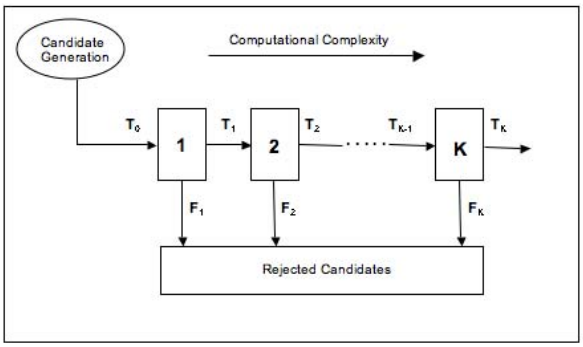

Figure 1. A general cascade framework used for online classification and offline training

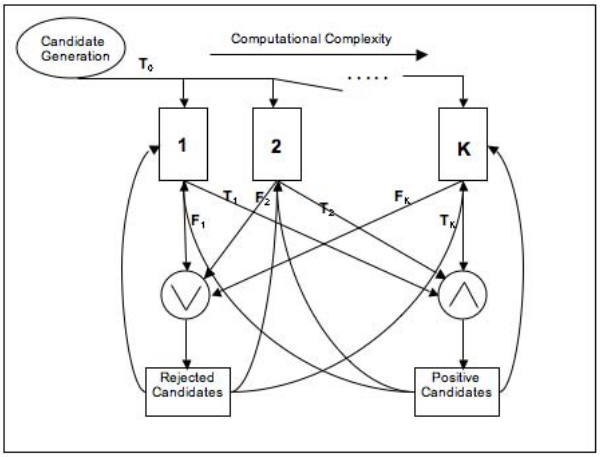

Figure 2. The proposed cascade framework for offline training of classifiers

hence screens out more true positives than necessary, then these true positives will never be recovered at later stages. The more stages in the cascade, the riskier the system becomes unstable. This observation motivates us to design a cascade that consists of significantly few stages. Furthermore, simple and low-cost image features are often not sufficient for detecting target structures especially in 3D medical images. Advanced features are indispensable for performance enhancement, but they require great computation time. If these features need to be calculated for a large portion of the candidates at the early stage of the cascade, the system may become prohibitively slow. Cascade AdaBoost treats all features equally when selecting features for each individual stage classifier, which leads to a computational inefficiency.

Unlike cascade AdaBoost, the recently proposed cascade approach for Computer Aided Detection (CAD) [1] incorporates the computational complexity of features into the cascade design. The cascading strategy in this approach brings some advantages: 1. High computational efficiency: early stages weed out many non-target patterns, so most stages are not evaluated for a typical negative candidate. Computationally expensive features are only calculated for a small portion of the candidates at later stages. 2. Robust system: the linear program with a ℓ_1-norm regularization at each stage is a robust system. Although no theoretical justification is derived, a cascade of very few stages unlikely harms the robustness of linear classifiers, opposed to a cascade of over 20 stages as AdaBoost cascade often obtains.

The fundamental problem with the cascade AdaBoost and other greedy cascade classifiers including the approach in [1] is that each classifier serves for its own purpose to optimize a local problem without worrying much about the overall performance. The cascade is trained by sequentially optimizing the classifiers at each stage. This training scheme potentially limits the performance of the overall system. For example there may be positive candidates in the dataset which would be incorrectly classified eventually at $stage\ K$. These candidates could be compromised for a bulk of negative candidates earlier in the cascade to gain significant computational advantages without sacrificing from the overall sensitivity. Likewise some of the positives which are missed in the earlier stages could have otherwise been detected in later stages with the help of additional features. Since the cascade is trained sequentially using the framework in Figure 1, cascade AdaBoost and other greedy cascade classifiers does not fully exploit the potential of a cascaded framework.

In this study we propose a cascade classification scheme, where the classifiers in the cascade are jointly trained to achieve optimization at the system level, i.e. maximize the area of the region of interest under the Receiver Operating Characteristics (ROC) curve at "stage K". In this approach the framework for online classification of candidates remains the same as in Figure 1 but the framework for offline training of the classifiers is modified as in Figure 2. The main motivation for our approach is that; a negative candidate is correctly classified by the cascade when at least one of the classifiers in the cascade rejects it whereas a positive candidate is classified correctly if all of the classifiers in the cascade detect it. The design parameters for an individual classifier may be updated depending on how well the other classifiers are performing. In this framework since we are optimizing at the system level some of the classifiers in the cascade may yield suboptimal performances.

3. Learning a Cascade Classifier in an AND-OR Framework

In this section we first review the hyperplane classifiers with hinge loss. Then develop the AND-OR training algorithm, which will form the basis for the proposed cascade framework.

3.1. Review of hyperplane classifiers with hinge loss

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.

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

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 strictly convex function. The weighted sum of strictly convex functions is also strictly convex. Therefore for a strictly convex function $\Phi(\alpha)$ (1) is also strictly convex. The problem in (1) can be formulated as a mathematical programming problem as follows:

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

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

3.2. Formulation of AND-OR learning

As discussed earlier, previous cascaded classification approaches train classifiers sequentially for each different stage, which amounts to a greedy scheme, meaning that the individual classifier is optimal only to the corresponding specific stage. The classifiers are not necessarily optimal to the overall structure where all stages are taken into account. The proposed approach optimizes all of the classifiers in the cascade in parallel by minimizing the regularized risk of the entire system and providing implicit mutual feedback to individual classifiers to adjust parameter design.

More specifically, we assume the feature vector $x_i \in \Re^d$ is partitioned into K subsets of non-overlapping variables in the order of increasing computational complexity as $x_i = (\bar{x}_{i1} \ldots \bar{x}_{iK})$, with $\bar{x}_{ik} \in \Re^{p_k}$ for $k = 1, \ldots, K$, $\sum_{k=1}^{K} p_k = d$. The feature set at $stage\ k$ is the accumulative set of all the k feature subsets i.e. $\acute{x}_{ik} = [\bar{x}_{i1} \ldots \bar{x}_{ik}]$ and $\acute{x}_{iK} = x_i$.

In the greedy scheme (Figure 1), the training set at stage k is $\{(\acute{x}_{ik}, y_i)\}$, for $i \in \Psi_{k-1}$, where Ψ_{k-1} is the set of indices of candidates classified positive by classifier at stage $k-1$, $\Psi_{k-1} = \{i : x_i \in T_{k-1}\}$. In this framework each classifier eliminates as many negative candidates as possible while satisfying the designated sensitivity and leave the remaining candidates to the next classifier in line, thus $\Psi_K \subset \Psi_{K-1} \subset \ldots \subset \Psi_0$. In the proposed framework as shown in Figure 2 all of the stages in the cascade are trained with the initial training dataset, i.e. $\Psi_k = \Psi_0 \forall k$.

We aim to optimize the following cost function

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

where $e_{ik} = 1 - \alpha_k^T y_i \acute{x}_{ik}$ and $(e_{ik})_+$ defines the hinge loss of the i-th training example $\{(\acute{x}_{ik}, y_i)\}$ induced by classifier k. Notice that classifier k uses a subset of the features in x_i so we denote the feature vector used by classifier k as $\acute{x}_{ik}$. 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 positive and negative samples. The loss induced by a positive sample is zero only if $\forall k : 1 - \alpha_k^T y_i \acute{x}_{ik} \leq 0$ which corresponds to the "AND" operation in Figure 2, and the loss induced by a negative sample is zero as long as $\exists k : 1 - \alpha_k^T y_i \acute{x}_{ik} \leq 0$ which corresponds to a "OR" operation.

The objective function in (3) is nonconvex and nonlinear, which by itself is computationally expensive to solve. In the next section we propose an efficient alternating optimization algorithm to solve this problem.

4. Cyclic Optimization of CCL

We develop an iterative algorithm which, at each iteration, carries out K steps, each aiming to optimize one clas-

sifier at a time. This type of algorithms is usually called alternating or cyclic optimization approaches. At any iteration, we fix all of the classifiers in the cascade but the classifier k. The fixed terms have no effect on the optimization of the problem once they are fixed. Hence solving (3) is equivalent to solving the following problem by dropping the fixed terms in (3):

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

where $w_i = \prod_{m=1, m\neq k}^{K} (e_{im})_+$.

This can be cast into a constrained problem as follows

$$\begin{array}{rrcl} \min\limits_{(\alpha_k,\xi_k)\in R^{d_k+\ell}} & \Phi_k(\alpha_k) & + & \nu_1 \sum_{i\in C^-} w_i\xi_i \\ & & + & \nu_2 \sum_{i\in C^+} \xi_i \\ \text{s.t.} & \xi_i & \geq & e_{ik},\ \forall i \\ & \xi_i & \geq & 0,\ \forall i \in C^- \\ & \xi_i & \geq & \gamma_i,\ \forall i \in C^+ \end{array} \tag{4}$$

where $\gamma_i = max\left(0, e_{i1}, \ldots, e_{i(m-1)}, e_{i(m+1)}, \ldots, e_{iK}\right)$. The subproblem in (4) 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 $w_i = \prod_{k=1,k\neq m}^{K} (e_{ik})_+$. This term multiplies out to zero for negative samples correctly classified by one of the other classifiers. For these samples $e_{im} < 0$ and $\xi_i = 0$ making the constraints on ξ_i in (4) redundant. As a result there is no need to include these samples when training the $stage - m$ of the cascade, which yields significant computational advantages. Second the lower bound for ξ is now $max\left(0, e_{i1}, \ldots, e_{i(m-1)}, e_{i(m+1)}, \ldots, e_{iK}\right)$. This implies that if any of the classifiers in the cascade misclassifies x_{im} the lower bound on ξ is no longer zero relaxing the constraint on x_{ik}.

4.1. An algorithm for AND-OR cascaded classifiers

(0) For each classifier in the cascade solve (2) for α_k^0 using the training dataset $\left\{\left(x_i^k, y_i\right)\right\}_{i=1}^{\ell}$ and initialize $\alpha_k = \alpha_k^0$, set counter $c = 1$ and maximum number of iterations to L.

(i) Fix all the classifiers in the cascade except classifier k and solve (4) 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 (3), for all $k = 1, \ldots, K$.

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

4.2. Convergence analysis

The cyclic optimization algorithm presented in Section 4.1 defines a point-to-set algorithmic mapping $\mathcal{A}$ which is actually a composite algorithmic mapping as $\mathcal{A} = \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_k \otimes \cdots \otimes \mathcal{A}_K$. Each sub-mapping $\mathcal{A}_k$ is characterized by the problem (4). We denote $\mathcal{A}_k(\hat{\alpha})$ as the algorithmic mapping specified by problem (4) where all α's except α_k are fixed to the corresponding values in $\hat{\alpha}$ and α_k is a variable to be determined. Denote $\Delta_k(\hat{\alpha})$ as the feasible region defined through the hinge loss constraints in problem (4) in terms of $\hat{\alpha}$ with only α_k free. Bear in mind that although omitted, problem (4) has other constraints defined by $\alpha_1, \cdots, \alpha_{k-1}, \alpha_{k+1}, \cdots, \alpha_K$ with fixed values. Hence we have

$$\begin{aligned}\mathcal{A}_k(\hat{\alpha}) \ = \ \text{argmin}\{&\mathcal{J}(\hat{\alpha}_1, \cdots, \hat{\alpha}_{k-1}, \alpha_k, \hat{\alpha}_{k+1}, \\ &\cdots, \hat{\alpha}_K) | \alpha_k \in \Delta_k(\hat{\alpha})\}\end{aligned} \tag{5}$$

where $\hat{\alpha}$ is a given specific value.

The algorithm in Section 4.1 can be re-stated as follows: at the c-th iteration, we start from the iterate $\alpha^c = (\alpha_1^c\ \alpha_2^c \cdots \alpha_K^c)$, and we solve

$$(\alpha_1^{c+1}\ \alpha_2^c \cdots\ \alpha_K^c) \ \in \ \mathcal{A}_1(\alpha^c) \tag{6}$$

$$(\alpha_1^{c+1}\ \alpha_2^{c+1} \cdots\ \alpha_K^c) \ \in \ \mathcal{A}_2(\alpha_1^{c+1}\ \alpha_2^c \cdots\ \alpha_K^c) \tag{7}$$

$$\cdots \ \in \ \cdots \tag{8}$$

$$(\alpha_1^{c+1}\ \alpha_2^{c+1} \cdots\ \alpha_K^{c+1}) \ \in \ \mathcal{A}_K(\alpha_1^{c+1}\ \alpha_2^{c+1} \cdots\ \alpha_K^c) \tag{9}$$

It is generally difficult to prove the convergence of a cyclic optimization algorithm to local minimizers. For many cyclic approaches, no convergence analysis can be substantially established. We prove that our algorithm at least converges from any initial point (global convergence) to the set of sub-optimal solutions. The solution $\hat{\alpha}$ is sub-optimal if the objective function $\mathcal{J}$ can not be further improved at $\hat{\alpha}$ following any directions defined by $\alpha_k,\ k = 1, \cdots, K$. In other words, the solution $\hat{\alpha}$ that our algorithm obtains satisfies: $\hat{\alpha}_k$ is a generalized fixed point of $\mathcal{A}_k$, i.e., $\hat{\alpha}_k \in \mathcal{A}_k(\hat{\alpha})$. The following theorem characterizes our result.

Theorem 4.1 *(a) The sequence $\{\mathcal{J}^c\}$ converges to a finite value $\hat{\mathcal{J}}$, (b) for any accumulation point of $\{\alpha^c\}$, denoted as $\hat{\alpha}$, if the operation of minimization over Δ_k occurs an infinite number of times in the subsequence which has limit $\hat{\alpha}$, then $\hat{\alpha}_k \in \mathcal{A}_k(\hat{\alpha})$.*

Proof. The proof is established by following the theorem by Fiotot and Huard [3, 8] which states as follows: Conclusions (a) and (b) hold if

- $\alpha \in \Delta_k(\alpha)$ for all α and k,
- Δ_s is upper-semicontinuous and lower-semicontinuous on Ω,
- if $\beta \in \Delta_k(\alpha)$, then $\Delta_k(\beta) = \Delta_k(\alpha)$,
- $\Omega_0 = \{\alpha \in \Omega | \mathcal{J}(\alpha) \leq \mathcal{J}(\alpha^0)\}$ is compact.

Obviously, α is itself feasible to $\Delta_k(\alpha)$. The only constraint we can have when we convert the unconstrained problem (3) into constrained subproblem (4) is the hinge loss $\alpha^T y_i x_i + \xi \geq 1$. The function $\alpha^T y_i x_i$ is a continuous function in terms of α, so the sets $\{\alpha | \alpha^T y_i x_i < constant\}$ and $\{\alpha | \alpha^T y_i x_i > constant\}$ are both open sets, proving Δ_k is upper-semicontinuous and lower-semicontinuous. If a point $\beta \in \Delta_k(\alpha)$, it means $\beta_m = \alpha_m$, for $m = 1, \cdots, K,\ m \neq k$. Then $\Delta_k(\beta) = \Delta_k(\alpha)$. The set Ω in our problem is the entire α space. For any initial point α^0, let $\mathcal{J}^0 = \mathcal{J}(\alpha^0)$. Then $\mathcal{J}(\alpha) \leq \mathcal{J}^0$ implies that $\sum_{k=1}^{K} \Phi_k(\alpha_k) \leq \phi$ which is a constant since the error terms (the second and third terms) in (3) are nonnegative. The set $\{\alpha | \sum_{k=1}^{K} \Phi_k(\alpha_k) \leq \phi\}$ defines a bounded and closed set, and is thus compact. ∎

5. CAD Applications

Although the proposed algorithm provides a general framework which can be employed for any classification purpose, the approach was highly motivated by the Computer-Aided Detection (CAD) system which detects abnormal or malignant tissues from medical imaging modality, such as CT, MRI and PET.

5.1. A CAD system: automatic polyp detection

Colorectal cancer is the third most common cancer in both men and women. It is estimated that in 2004, nearly 147,000 cases of colon and rectal cancer will be diagnosed in the US, and more than 56,730 people would die from colon cancer [7], accounting for about 11% of all cancer deaths. In over 90% of the cases colon cancer progressed rapidly is from local (polyp adenomas) to advanced stages (colorectal cancer), which has very poor survival rates. However, identifying (and removing) lesions (polyp) when still in a local stage of the disease, has very high survival rates [2], thus illustrating the critical need for early diagnosis.

In the area of colon cancer detection from CT images, the key aspect of a CAD system is the improvement in sensitivity with respect to detection of polyps that such system can offer. In order to put the role of colon CAD as a second reader in proper context, we outline it in the context of a complete clinical workflow.

A workflow with integrated Colon CAD system would consist of the following 4 stages:

- Case Loading: physician loads the case for review - CAD system begins processing in the background.
- First read: physician reviews the case, prone and supine, finalizes its findings.
- CAD results are invoked (CAD button is pressed): physician acknowledges she/he has completed the review of the case.
- Second read: physician reviews additional CAD findings, and rejects any considered false positives.

To avoid any delays in the workflow of a physician the CAD results should be ready by the time physician completes the first read. Therefore there is a run-time requirement a CAD system needs to satisfy. The stages involved during online processing of a volume and the median time it takes are listed below.

- Interpolation, 47 sec
- Detagging, 50 sec
- Segmentation, 38 sec
- Candidate Generation, 1 min 29 sec
- Feature Computation, 4 min 46 sec
- Classification, negligible
- Total, 8 min 14 sec

Feature computation is by far the most computational stage of online processing (4 min 46 sec). In this study we aim to speed up this stage by first partitioning the original feature set into subsets with increasing computational costs. Then we use the proposed AND-OR framework to build a cascade classifier in which earlier stages are trained by the less computational features. This way during online execution earlier stages filter out as many candidates as possible before later stages that require advanced features are called upon.

5.2. Numerical Results

We validate the proposed cascade classification algorithm with respect to its generalization performance and computational efficiency. We compared our algorithm to a single stage SVM classifier constructed using all the features, and the commonly-used cascade classifier based on AdaBoost.

5.2.1 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=169) and test (n=201). The test group was sequestered and only used to evaluate the performance of the final system.

Training Data Patient and Polyp Info: There were 169 patients with 338 volumes. The candidate generation (CG) algorithm identifies a total of 88 polyps while generating an average of 137.7 false positives per volume. *Testing Data Patient and Polyp Info:* There were 201 patients with 396 volumes. The candidate generation (CG) algorithm identifies a total of 106 polyps while generating an average of 139.4 false positives per volume.

The candidate generation algorithm was independently applied to the training and test sets, achieving 90.72% detection rate on the training set at 137.7 FPs per volume and 90.91% detection rate on the test set at 139.4 FPs per volume, resulting in totally 46764 and 55497 candidates in the respective training and test sets.

Numerical image features of totally 46 were designed, 7 of which came from the CG step with no extra computational burden. The remaining features were grouped into three according to their computational costs. A three-stage cascade classifier was built. The feature set 1 which was composed of the least computational 17 features took 0.2 sec cpu time per candidate, and was used together with the CG features to train the first classifier. The feature set 2 which was composed of 7 relatively more computational features required an average of 0.5 sec. cpu time per candidate, and was used together with the feature set 1 and CG features to learn the second classifier. The feature set 3 which was composed of 15 computationally most demanding features required on average of 1.4 sec. cpu time per candidate, and was used with all the other features to learn the third classifier.

5.2.2 Classifier design

We set aside 30% of the training data as validation data and use it for parameter tuning. During this process the classifiers are trained with the 70% and are validated with the remaining 30%. However when the tuning process is over, i.e. classifier parameters, thresholds are all set, the entire training data is used to train the classifiers. In most CAD applications 0-5 fp/vol is defined as the clinically admissible range of false positives per volume. Therefore this region on the ROC curve is set as our region of interest (ROI) and all classifiers are tuned to maximize this portion of the area under the ROC curve with the validation data.

For the proposed approach we set $\Phi_k(k) = \|\alpha\|_2^2$, i.e. SVM. The parameters ν_1 and ν_2 are estimated from a coarsely tuned discrete set of five values. The values that maximizes the area of ROI under the ROC curve for the overall system with the validation dataset are found as $\nu_1 = 50$ and $\nu_2 = 1$.

In the experiments with cascade AdaBoost, we referred to the procedure described in [10][1]. Unlike [10] where each stage classifier was learned in the presence of all the available features, similar to the design of the proposed cascaded classifier we split the feature set into three and adopt a three phase training of the AdaBoost cascade. During Phase 1 using the feature set 1 and the CG features the first k_1 stages of the cascade were built. The feature set 2 is used together with feature set 1 and CG features to build the stages $k_1 + 1$ through k_2 of the cascade in Phase 2. Finally during Phase 3 all of the features were used and stages $k_2 + 1$ through k_3 were built. The validation dataset was used to tune the decision thresholds and to estimate the number of stages in each phase, i.e. k_1, k_2, k_3. The decision thresholds at each stage were tuned to satisfy the target sensitivity for each phase. Based on the performance with the validation set the desired target sensitivities were no polyp miss for phase one and two and one polyp miss per stage for phase three of the training. The number of stages in each phase were estimated to minimize the area of ROI under the ROC curve. During each phase the number of stages is increased in a greedy manner until no further increase in this area is observed upon when the next phase starts. Designing the cascade AdaBoost classifier this way yielded one stage for phase 1, one stage for phase 2 and three stages for phase 3 making a five-stage cascade classifier.

We also designed an SVM classifier using all of the features available without any cascading. The parameters $\nu_1 = 15$ and $\nu_2 = 2$ are estimated using the same approach described above for the proposed AND-OR cascade.

5.2.3 Generalization performance and speed

The three classifiers obtained respectively by AND-OR cascade, single stage SVM, and cascade AdaBoost were evaluated on the sequestered test set. The results that includes the total number of polyps detected and the remaining number of candidates per volume after each stage in the cascade can be found in Table 1. Also present in this table is the average feature computation time per volume in CPU seconds. Overall system performance at 5fp/vol for the three classifiers is also included in this table. Finally the overall system performance obtained by the three classifiers are compared in Figure 3 by plotting the ROC curves corresponding to the final stage of the cascade classifiers and single-stage SVM.

[1]The AdaBoost approach [6] used in [10] has been implemented by other sources. A MatLab version of the implementation was downloaded from MatLab Statistics web page *http://www.mathtools.net/MATLAB/Statistics/*

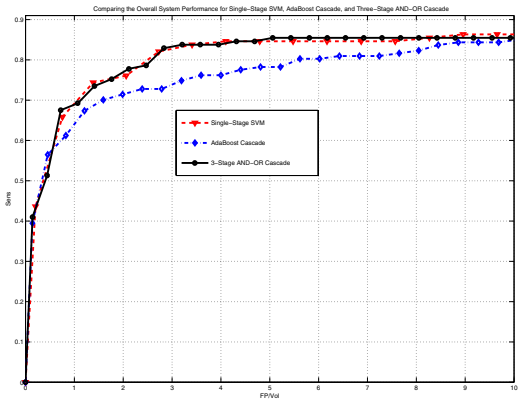

Figure 3. ROC curves obtained by the three classifiers on the test data.

Clearly, both cascade classifiers demonstrate computational efficiency compared to single-stage SVM. The AND-OR cascade requires an average of 81.0 CPU secs per volume for feature computation across the three stages, AdaBoost cascade requires 118 CPU secs for feature computation for the three phases (five stages). Both numbers are significantly lower than the 286.0 cpu secs required for the single-stage SVM with the proposed AND-OR cascade being more than three times faster than the single-stage SVM. From a performance generalization perspective we observe no statistically significant difference between the proposed cascade classifier and the single stage SVM when the ROC curves obtained by these classifiers on the test data are compared. When the proposed AND-OR Cascade is compared against the AdaBoost Cascade we achieve roughly 30% improvement in the average online processing time of a volume and observe statistically significant improvement in the performance with a p-value=0.002 in favor of the AND-OR cascade (p-value is computed for the ROI, using the technique in [5]). The main motivation for our work was to speed up the online execution of the CAD system without sacrificing from the system performance. This is achieved through the proposed AND-OR cascade algorithm by constructing a series of linear classifiers all trained simultaneously in an AND-OR framework using features of increasing complexity.

6. Conclusions

We have proposed and investigated a novel cascade classification algorithm based on an AND-OR design of hyperplane classifiers. The proposed approach aims to address the shortcomings of previous approaches where each classifier in the cascade is optimized locally. Our approach optimizes the overall system performance even if this requires ending up with non-optimal local classifiers. We also provide a complete implementation to solve the proposed classification formulation through alternating optimization techniques. Preliminary convergence analysis is further supplied to justify our algorithm and our algorithm converges to a cascade of classifiers where each classifier in the cascade is a fixed point of the algorithmic mapping, and hence optimal when the cascade is fixed in any other stages.

The content of this paper focuses on the offline training of the classifiers which does not necessarily restrict how the classifiers should be ordered in the cascade sequence. The order of the classifiers is problem-specific in our cascade design. Any suitable domain or prior knowledge can be used to determine it. For example, in our experiments, the computational complexity of different sets of features should be a major factor in deciding which classifiers need to be executed earlier, and which later. Our experiments show favorable results in comparison with single-stage classifiers and the cascade AdaBoost. The proposed framework significantly reduces the online execution time of the CAD system as illustrated in the system of automatic polyp detection.

References

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

[2] L. Bogoni, P. Cathier, M. Dundar, A. Jerebko, S. Lakare, J. Liang, S. Periaswamy, M. Baker, and M. Macari. Cad for colonography: A tool to address a growing need. *British Journal of Radiology*, 78:57–62, 2005.

[3] J. C. Fiorot and P. Huard. Composition and union of general algorithms of optimization. *Mathematical Programming Study*, 10:69–85, 1979.

[4] Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

[5] J. A. Hanley and B. J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148:839–843, 1983.

[6] T. H. J. Friedman and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38, 2000.

[7] D. Jemal, R. Tiwari, T. Murray, A. Ghafoor, A. Saumuels, E. Ward, E. Feuer, and M. Thun. Cancer statistics, 2004.

[8] J. D. Leeuw. Block relaxation algorithms in statistics, 1994.

[9] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37, 1999.

[10] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57, 2004.

	AND-OR Cascade			SVM	Cascade AdaBoost				
					Phase 1	Phase 2	Phase 3		
	C_1	C_2	C_3		C_1	C_2	C_3	C_4	C_5
Avg. CPU time per volume (sec.)	28	27.5	25.48	294.0	26.0	25.0	67.48		
number of polyps found (out of 106)	105	100	100	99	103	103	102	102	91
False positives per volume (initially 139.4)	55	18.2	5	5	50	48.2	30.8	20.6	5.0

Table 1. Results by the three classifiers obtained, respectively, by our approach, the traditional SVM, and the cascade AdaBoost. Numbers are given for every stage in the cascade.

[11] T. S. X. Tang, Z. Ou and P. Zhao. Cascade adaBoost classifiers with stage features optimization for cellular phone embedded face detection. *Lecture Notes in Computer Science (Advances in Natural Computation)*, 2005.