Designing Efficient Cascaded Classifiers: Tradeoff between Accuracy and Cost

Vikas C. Raykar
CAD and Knowledge Solutions
Siemens Healthcare
Malvern, PA 19355 USA
vikas.raykar@siemens.com

Balaji Krishnapuram
CAD and Knowledge Solutions
Siemens Healthcare
Malvern, PA 19355 USA
balaji.krishnapuram@siemens.com

Shipeng Yu
CAD and Knowledge Solutions
Siemens Healthcare
Malvern, PA 19355 USA
shipeng.yu@siemens.com

ABSTRACT

We propose a method to train a cascade of classifiers by simultaneously optimizing all its stages. The approach relies on the idea of optimizing soft cascades. In particular, instead of optimizing a deterministic hard cascade, we optimize a stochastic soft cascade where each stage accepts or rejects samples according to a probability distribution induced by the previous stage-specific classifier. The overall system accuracy is maximized while explicitly controlling the expected cost for feature acquisition. Experimental results on three clinically relevant problems show the effectiveness of our proposed approach in achieving the desired tradeoff between accuracy and feature acquisition cost.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*; I.5.2 [**Pattern Recognition**]: Models—*Statistical*; H.2.8 [**Database Applications**]: Data mining; I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

cascade design, cost sensitive learning, accuracy vs cost

1. INTRODUCTION

In many applications features are *acquired on demand*; usually a set of features can be acquired as a group. However each feature group incurs a certain cost. This cost could be either computational, financial, or human discomfort. For example, in certain medical applications (see Section 7) some tests are very expensive (e.g. acquiring some blood biomarkers or a MRI) or cause extreme discomfort (e.g. biopsy) for the patient.

This motivates the design of a *cascade of classifiers*–each stage using features with *increasing predictive power* and also *increasing acquisition cost.* Each stage of the cascade can either accept and pass a sample into the next stage for further feature acquisition and further classification analysis or it can reject the sample immediately–classifying it as a negative class sample–thus avoiding any further (downstream) feature acquisition cost. Typically we would like the first stage of our classifier to use the cheapest features and the most expensive features at the later stage.

In general, cascaded classifier design is used to reduce run time of the overall classifier by reducing the number of samples which need the computation of more expensive features. However, this process of cascaded classification may reduce accuracy of classification somewhat, but at the same time we do not want to sacrifice accuracy too much. In this paper we describe a method to *jointly train all the stage-wise classifiers* of a cascade of classifiers in order to maximize classification accuracy while simultaneously restricting the runtime explicitly. We develop principled methods to achieve the *desired tradeoff between accuracy and cost.*

1.1 Previous work and Novel Contributions

Most previous work on designing cascades are based on the Viola-Jones cascade framework [10] and are developed for building rapid real-time object detection systems (see [10, 1, 12] and references therein). Each stage of the cascade is an Adaboost classifier with decisions stump as the base learner and has its own threshold. The Viola-Jones cascade was developed in the context of face detection where we have a large number of very cheap features–all of them having the same relatively low computational cost. There is no explicit notion of feature groups and different acquisition costs. Feature cost has also been considered in the framework of cost-sensitive learning [7, 6, 9].

This paper makes two novel contributions. First we propose an algorithm to jointly train all the stages in a cascade. Second we provide a knob to control the tradeoff between accuracy and cost.

1. **Joint training of all stages:** Conventionally a cascade is trained in a *sequential manner* [10, 1, 12]. Only examples for which the classifier score is greater than a certain threshold pass through the next stage of the cascade. Each stage of the classifier is trained using *only* those examples which pass through all the previous stages. This is clearly not the optimal solution. Also in sequential training we have to focus on opti-

KDD'10, July 25–28, 2010, Washington, DC, USA.

mally choosing these thresholds to maximize a certain performance metric, since the training process depends on the choice of the thresholds.

Starting with logistic regression as the base classifier for each stage we propose a method to *jointly train a cascade* of classifiers. This is achieved by relaxing the hard cascade into a *soft cascade* as described in Section 3. Since we relax our cascade into a soft one we can train the entire cascade jointly and choose the thresholds as a post-processing step. Our usage of the term soft cascade is very different from the one proposed in [1]. A related recent paper [4] also proposes to jointly train a cascade of SVMs using an AND-OR framework. Our proposed solution is probabilistic and more importantly they do not take the costs into consideration.

2. **Tradeoff between accuracy and cost:** The expected cost to acquire the features is explicitly incorporated in the optimization problem (see Section 5). In some applications the cost may be a deciding factor and the user may be willing to sacrifice some accuracy. The proposed method for cascade training can be tuned to reflect this tradeoff between cost and accuracy. We are not aware of any previous work which tries to explicitly measure or minimize run time although they may implicitly assume that cascaded classifier design reduces run time by its very nature.

3. **Computation cost of training** Since we relax our cascade into a soft one we have decoupled the threshold selection from the training process. The entire cascade has to be trained jointly only once and the thresholds can be chosen as a post-processing step. In contrast for sequential training for each choice of the thresholds we have to retrain the cascade. This can be very expensive if the number of stages is large and the threshold selection is done over a fine grid. For example for a five stage cascade if we have to search over 10 thresholds for each stage in sequential training we have to train the cascade roughly 10^5 times compared to just once for the proposed joint training.

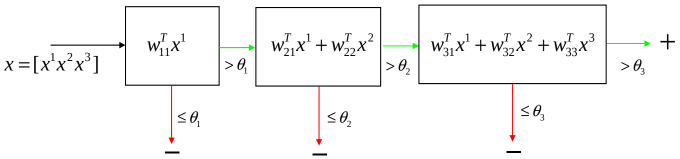
The rest of the paper is organized as follows. In Sections 2 and 3 we discuss how a hard cascade can be relaxed into a soft cascade for joint training. In Section 4 we describe a method to jointly train all the stages in the cascade via maximum a-posteriori estimation. The expected feature acquisition cost is explicitly modeled in Section 5, thus providing us with a knob to achieve the desired tradeoff between accuracy and cost. Experimental results on three medical datasets are discussed in Section 7.

2. CASCADE OF CLASSIFIERS

In a typical supervised learning scenario for binary classification we are given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ containing N instances, where $\mathbf{x}_i \in \mathcal{X}$ is an instance and $y_i \in \mathcal{Y} = \{0, 1\}$ is the corresponding known label. The task is to learn a classification function $f : \mathcal{X} \rightarrow \mathcal{Y}$. Typically an instance is represented as a d-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^d$.

2.1 Single stage linear classifier

We consider the family of linear discriminating functions: $\mathcal{F} = \{f_{\mathbf{w}}\}$, where for any $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$, $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. The final binary classifier can be written in the following form $y = 1$ if $\mathbf{w}^\top \mathbf{x} > \theta$ and $y = 0$ if $\mathbf{w}^\top \mathbf{x} \leq \theta$. The *threshold parameter* θ determines the operating point of the classifier. The Receiver Operating Characteristic (ROC) curve is obtained as θ is swept from $-\infty$ to ∞. Learning a *single stage* classifier implies choosing the weight vector $\mathbf{w}$ given the training data $\mathcal{D}$.



(c) Scenario 3

Figure 1: *A cascade of three linear classifiers.* (a) In scenario 1 each stage of the cascade uses only a subset of the features. (b) In scenario 2 each stage can also use the features from all the previous stages since they are already computed. (c) In scenario 3 the weights for each stage are separate and not shared with the previous stage as in scenario 2.

2.2 Cascade of linear classifiers

We will denote a K stage cascade by $[\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K]$. The features for any instance $\mathbf{x} \in \mathbb{R}^d$ is divided into K distinct sets– $\mathbf{x} = \left[\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^K\right]$. This feature grouping is usually done based on the cost required to acquire these features. Let t_j be an estimate of the cost it takes to acquire/compute feature subset $\mathbf{x}^j$. Typically the cascade is ordered by the feature acquisition time t_j, *i.e.*, would like the first stage of our classifier to use the cheapest features and the most expensive features at the later stage. We will describe three different scenarios in which the cascade can typically operate (see Figure 1). Our proposed training procedure easily works with all these scenarios.

- **Scenario 1** Each stage $\mathcal{C}_j$ of the cascade uses only the subset $\mathbf{x}^j$ of the features. Each cascade $\mathcal{C}_j$ is from a family of linear discriminating functions, where for any $\mathbf{x} \in \mathbb{R}^d$, $f_{\mathcal{C}_j}(\mathbf{x}) = \mathbf{w}_j^\top \mathbf{x}^j$.

- **Scenario 2** Each stage $\mathcal{C}_j$ uses the subset $\left[\mathbf{x}^1, \ldots, \mathbf{x}^j\right]$ of the features, *i.e.*, the stage j can also use the features from all the previous stages since they are already computed. Each cascade $\mathcal{C}_j$ is from a family of linear discriminating functions, where for any $\mathbf{x} \in \mathbb{R}^d$ $f_{\mathcal{C}_j}(\mathbf{x}) = \sum_{k=1}^{j} \mathbf{w}_k^\top \mathbf{x}^k$.

- **Scenario 3** This is same as the previous scenario. But an important difference is that the weights for each stage are separate and not shared with the previous stage as in scenario 2. Hence each cascade $\mathcal{C}_j$ is from a family of linear discriminating functions, $f_{\mathcal{C}_j}(\mathbf{x}) = \sum_{k=1}^{j} \mathbf{w}_{jk}^\top \mathbf{x}^k$.

Each stage predicts $y = 1$ if $f_{\mathcal{C}_j}(\mathbf{x}) > \theta_j$ and $y = 0$ if $f_{\mathcal{C}_j}(\mathbf{x}) \leq \theta_j$ where θ_j is the threshold parameter for each stage. Given the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ containing N instances we have to estimate the weight vector $\mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K]$ and choose the thresholds for each stage $\theta = [\theta_1, \ldots, \theta_K]$ to minimize the error and also the average cost. The sequential design of a traditional hard cascade needs the thresholds $\theta_1, \ldots, \theta_{j-1}$ for previous stages to be fixed before designing the next stage classifier parameters. Each stage of the classifier is trained using *only* those examples which pass through all the previous stages.

2.3 Logistic generalized linear model

In order to train our classifier we will use the logistic regression model. For each stage the posterior probability for the positive class is written as a logistic sigmoid acting on the linear classifier f_j, *i.e.*,

$$p_{\mathcal{C}_j}(y = 1|\mathbf{x}, \mathbf{w}) = \sigma\left(f_{\mathcal{C}_j}(\mathbf{x})\right). \quad (1)$$

The logistic sigmoid function–also known as the squashing function–is defined as $\sigma(z) = 1/(1+e^{-z})$. This classification model is known as *logistic regression.* Since we are dealing with a binary classification problem $p_{\mathcal{C}_j}(y = 0|\mathbf{x}, \mathbf{w}) = 1 - \sigma\left(f_j(\mathbf{x})\right)$.

3. SOFT CASCADE

If we directly incorporate the thresholds, then during training we have to solve a discrete optimization problem which is not easy. Although the eventual test set will have to be evaluated using a hard cascade which explicitly thresholds and thus rejects a subset of samples at each stage, we propose to *design* the parameters of this system by instead optimizing a surrogate cascade of soft-classifiers. Rather than a hard rejection, the stages of the surrogate system optimized during training only provide a probability that the sample is negative. In intuitive physical terms, this soft cascade may be viewed as stochastically rejecting a sample at any stage based on the posterior class probability evidenced by the classifier for that stage.

For the overall soft cascade system, an instance $\mathbf{x}$ is classified as positive if *all* the K stages in the cascade predict it as positive. The probability that all stages predict it as positive can be written as

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \prod_{j=1}^{K} p_{\mathcal{C}_j}(y = 1|\mathbf{x}, \mathbf{w}) = \prod_{j=1}^{K} \sigma\left(f_{\mathcal{C}_j}(\mathbf{x})\right). \quad (2)$$

An instance $\mathbf{x}$ is classified as negative if *at least one* of the K classifiers predicts it as negative. Hence

$$p(y = 0|\mathbf{x}, \mathbf{w}) = 1 - \prod_{j=1}^{K} \sigma\left(f_{\mathcal{C}_j}(\mathbf{x})\right). \quad (3)$$

Note that the hard and soft cascades demonstrate somewhat different properties. For example, *the sequential ordering of the cascade is not important for a soft cascade*, although it certainly matters for the hard cascade. In other words, even if we switch the order of the stages of the soft cascade the overall accuracy is not affected, but the ordering of the cascade is crucial when considering the cost. This relaxation is a device to ease the training process and as such the order definitely matters during testing.

Nevertheless, in order to optimize all stages of a cascade simultaneously to optimize accuracy we need a strategy that allows us to model the inter relationships between the stages. In other words, we want a mathematical method that explicitly accounts for the fact that it is sufficient for a sample to be rejected at any stage of the classifier, we do not have to force multiple stages to reject it. This means that the joint design of cascades can potentially allow each stage to focus on a different type of false positive and thus improve overall accuracy as compared to a traditional one-stage-at-a-time cascade design which ignores this information. In order to utilize this intuition, we propose to design the cascade by optimizing a soft cascade even though the final test set will be evaluated by a hard cascade (in order to be able to reduce the run time). Having relaxed the classifier into a soft one we now consider how to train the entire cascade jointly.

4. TRAINING THE CASCADE

The maximum likelihood estimate for $\mathbf{w}$ is given by

$$\begin{aligned} \widehat{\mathbf{w}}_{\mathrm{ML}} &= \arg\max_{\mathbf{w}} p(y_1, \ldots, y_N|\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \log p(\mathcal{D}|\mathbf{w}). \end{aligned} \quad (4)$$

Define $p_i = p(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \prod_{j=1}^{K} \sigma\left(f_{\mathcal{C}_j}(\mathbf{x}_i)\right)$–the probability that the i^{th} instance $\mathbf{x}_i$ is positive. Assuming that the training instances are independent the log-likelihood can be written as

$$l(\mathbf{w}) = \log p(\mathcal{D}|\mathbf{w}) = \sum_{i=1}^{N} y_i \log p_i + (1 - y_i) \log(1 - p_i). \quad (5)$$

The ML solution in practice can exhibit severe over-fitting especially for high-dimensional data. This can be addressed by using a prior on $\mathbf{w}$ and then finding the *maximum a-posteriori* (MAP) solution. In order to promote sparsity we impose a Laplace prior (with a common scale parameter γ) on each parameter w_i.

$$p(w_i|\gamma) = \frac{\sqrt{\gamma}}{2} \exp\left(-\sqrt{\gamma}|w_i|\right). \quad (6)$$

We also assume that individual weights in $\mathbf{w}$ are independent and hence the overall prior is the product of the priors for each component.

$$p(\mathbf{w}|\gamma) = \prod_{i=1}^{d} p(w_i|\gamma) = \left(\frac{\sqrt{\gamma}}{2}\right)^d \exp\left(-\sqrt{\gamma}\|\mathbf{w}\|_1\right), \quad (7)$$

where $\|\mathbf{w}\|_1 = \sum_{i=1}^{d} |w_i|$ is the l_1-norm.

Once we observe the training data $\mathcal{D}$ we will update the prior to compute the posterior $p(\mathbf{w}|\mathcal{D})$, which can be written as follows (using Bayes's rule)–

$$p(\mathbf{w}|\mathcal{D}, \gamma) = \frac{p(\mathcal{D}|\mathbf{w}) p(\mathbf{w}|\gamma)}{\int p(\mathcal{D}|\mathbf{w}) p(\mathbf{w}|\gamma) d\mathbf{w}}. \quad (8)$$

This posterior can then be used to compute predictive distributions, which will typically involve high dimensional integrals. For computational efficiency we will base our prediction on point estimates of $\mathbf{w}$. We could either use the mean, median, or the mode of the posterior. However the posterior is difficult to compute because of the integral in the denominator. Hence we use the mode of the posterior, since the denominator does not depend on $\mathbf{w}$. The mode of the posterior–the *maximum a-posteriori* (MAP) estimate is given by

$$\begin{aligned}\widehat{\mathbf{w}}_{\mathrm{MAP}} &= \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D},\gamma) \\ &= \arg\max_{\mathbf{w}} \left[\log p(\mathcal{D}|\mathbf{w}) + \log p(\mathbf{w}|\gamma)\right]. \quad (9)\end{aligned}$$

Substituting for the log likelihood (Eq. 5) and the prior (Eq. 7) we have

$$\widehat{\mathbf{w}}_{\mathrm{MAP}} = \arg\max_{w} L(\mathbf{w}), \quad (10)$$

where

$$L(\mathbf{w}) = \left[\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i)\right] - \sqrt{\gamma}\|\mathbf{w}\|_1. \quad (11)$$

The terms which do not depend on $\mathbf{w}$ have been omitted out.

5. MODELING THE EXPECTED COST

An crucial motivation for adopting cascade structure is that we have limited cost. Hence we would like to find the MAP estimate subject to the constraint that the expected cost for a new instance

$$E_{p(\mathbf{x})}\left[\mathcal{T}(\mathbf{x})\right] \leq c, \quad (12)$$

where $\mathcal{T}$ is the cost for a new instance $\mathbf{x}$. The expectation is over the unknown test distribution. Since we do not know $p(\mathbf{x})$ we can use an estimate of this quantity based on the training set.

Consider a training instance $\mathbf{x}_i$. The first stage takes cost t_1 to compute the set of features $\mathbf{w}_1$. Once the features are computed it declares it as positive with probability $\sigma\left(f_{\mathcal{C}_1}(\mathbf{x}_i)\right)$. This means that $\mathbf{x}_i$ passes through to the second stage of the cascade with probability $\sigma\left(f_{\mathcal{C}_1}(\mathbf{x}_i)\right)$. The second stage now takes cost t_2 to acquire the set of features $\mathbf{w}_2$. The second stage declares it a positive with probability $\sigma\left(f_{\mathcal{C}_2}(\mathbf{x}_i)\right)$. Hence it passes through to the third stage of the cascade with probability $\sigma\left(f_{\mathcal{C}_1}(\mathbf{x}_i)\right)\sigma\left(f_{\mathcal{C}_2}(\mathbf{x}_i)\right)$. So given the parameters $\mathbf{w}$ an estimate of the expected cost can be written as

$$\mathcal{T}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left[t_1 + \sum_{j=2}^{K} t_j \prod_{l=1}^{j-1} \sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right)\right]. \quad (13)$$

So the optimization problem is

$$\widehat{\mathbf{w}}_{\mathrm{MAP}} = \arg\max_{w} L(\mathbf{w}) \quad \text{subject to} \quad \mathcal{T}(\mathbf{w}) \leq c. \quad (14)$$

In practice we solve the following unconstrained optimization problem

$$\widehat{\mathbf{w}}_{\mathrm{MAP}} = \arg\max_{w} L(\mathbf{w}) - \beta\mathcal{T}(\mathbf{w}), \quad (15)$$

where β controls the tradeoff between accuracy and cost.

6. THE OPTIMIZATION PROBLEM

The final optimization problem can now be written as

$$\widehat{\mathbf{w}} = \arg\min_{w} J(\mathbf{w}) \quad (16)$$

where

$$J(\mathbf{w}) = -l(\mathbf{w}) + \alpha\|\mathbf{w}\|_1 + \beta\mathcal{T}(\mathbf{w}), \quad (17)$$

where $l(\mathbf{w})$ is the log-likelihood that measures the accuracy, $\alpha = \sqrt{\gamma}$ controls the amount of sparsity, and β controls the cost.

In order to minimize $J(\mathbf{w})$ we use the cyclic coordinate descent algorithm because of its simplicity, speed, and stability. Methods of this flavor has been earlier used for lasso penalized regression problems [11] and logistic regression [13, 5]. We first set all the parameters to some initial value (zero). It sets the first variable to a value that minimizes the objective function, holding all other parameters constant. The algorithm then cycles through all the parameters and updates them in turn. Multiple passes are made until some convergence criterion is met. The Newton-Raphson update for the one-dimensional minimization problem is given by

$$w_t^{\mathrm{new}} = w_t + \eta\Delta w_t, \quad (18)$$

where η is the step-length and Δw_t is the Newton update given by

$$\Delta w_t = -\frac{J^{'}(w_t)}{J^{''}(w_t)}. \quad (19)$$

In order to avoid large updates we use a trust region $\Delta_t > 0$ which $|\Delta w_t|$ is not allowed to exceed on a single iteration[13, 5], *i.e.*,$\Delta w_t \leftarrow \min(\max(\Delta w_t, -\Delta_t), \Delta_t)$. The trust region is adapted every pass using $\Delta_t^{\mathrm{new}} = \max(2|\Delta w_t|, \Delta_t/2)$. We update $w_t^{\mathrm{new}} = w_t + \eta\Delta w_t$ only once before going to the next parameters. The componentwise derivatives and the convergence criterion are derived in the appendix.

7. EXPERIMENTS

We evaluate the proposed algorithm on three clinically relevant proprietary medical datasets where acquisition cost plays an important role.

- **Survival Prediction for Lung Cancer** Our first dataset concerns the 2-year survival prediction for advanced non-small cell lung cancer (NSCLC) patients treated with chemo/radiotherapy. The task is to predict whether the patient will survive for more than 2 years. We consider four groups of features which are known to be predictive for this problem [3], with *increasing predictive power* and also *increasing acquisition cost*:

  - 9 *Clinical features* such as gender, age, etc.
  - 8 *features from tests before therapy* such as lung function, creatinine clearance, etc.
  - 7 *imaging and treatment features* such as gross tumor volume, treatment dose, etc.
  - 21 *Blood bio-markers* such as Interleukin-8, Osteopontin, etc..

  The cost to acquire these features is given as 0, 1, 2, and 5 respectively. The clinical features are already available and have zero acquisition cost while acquiring the blood bio-markers is costly. The study [3] contains 82 advanced NSCLC patients treated at the MAASTRO Clinic in the Netherlands from 2002 to 2006, among which 24 survived 2 years (hence positive instances for training).

- **Pathological Complete Response (pCR) Prediction for Rectal Cancer** Our second example is to predict tumor response after chemo/radiotherapy for locally advanced rectal cancer. This is very important in individualizing treatment strategies, since patients with a pCR after therapy, *i.e.*, with no evidence of viable tumor on pathologic analysis, would need less invasive surgery or another radiotherapy strategy instead of resection. Most available models combine clinical factors such as gender and age, and pretreatment imaging-based factors such as tumor length and SUV$_{max}$ (from CT/PET imaging), but it is expected that adding imaging data collected after therapy would lead to a better predictive model (with a higher cost certainly). We use data from [2] which contains 78 prospectively collected rectal cancer patients. 21 of them had pCR. All patients underwent a CT/PET scan before treatment and 42 days after treatment. We have 2 groups of features:

  – 8 features based on clinical information and CT/PET scan before treatment, and

  – 2 features based on the difference of CT/PET scans before and after treatment.

  The cost for the second feature group (assigned to 10) is much higher than the first group (assigned to 1) because a CT/PET scan is an expensive procedure.

- **Computer aided diagnosis of lung cancer** Lung cancer is a leading cause of cancer related death in western countries. With the advent of computed tomography (CT) and computer aided detection (CAD) systems [8] it is now possible to detect pulmonary nodules (which are usually precursors to cancer) during early stages leading to early intervention. A CAD system aids the radiologist by marking the location of likely nodules on a CT scan. Most CAD algorithms operate in a sequence of three stages–(1) *Candidate generation*–this step identifies potentially unhealthy regions of interest. While this step can detect most of the anomalies, the number of false positives will be extremely high. (2) *Feature computation*–computation of a set of descriptive morphological features for each of the candidates. (3) *Classification*–labeling of each candidate as a nodule or not by a classifier. Based on a set of features computed for these candidates we can train a classifier which can discriminate a nodule from other candidates. However we want the run time of the classifier during testing be as small as possible. This motivates the proposed cascade design. A certain image processing step computes a group of features. Each group requires a different amount of time to compute them. In this experiment we use a set of three feature groups. The number of features are 9, 23, and 15 for these three groups. The average cost to compute these features are 1.07, 3.10, and 20.7 seconds respectively. For training we used 196 CT scans with 923 positive candidates and 54455 negative candidates. The performance was verified on an independent test set containing 113 CT scans with 585 positive candidates (277 nodules) and 32977 negative candidates.

7.1 Methods compared

We compare the results for the following methods.

1. **Single stage classifier** This corresponds to a single classifier trained by acquiring all the features at once. This is essentially a single stage cascade (which for our model is a sparse logistic regression classifier) and acts as our baseline. The average cost for this model is one, since it uses all the features.

2. **Proposed soft cascade** $\beta = 0$ The proposed method of jointly training the cascade. While the proposed method can be used for all the scenarios described in section 2 we report results only for scenario 1 where each stage uses only a disjoint set of features. We found that for the datasets in this paper all three scenarios gave similar performance. We also set $\beta = 0$ so that we do not explicitly account for the cost of feature acquisition during training. The l_1-sparsity parameter α was chosen based on a 5-fold cross-validation on the training set.

3. **Sequential Training: Logistic Regression** The proposed cascade classifier trained sequentially, *i.e.*, each stage of the classifier is trained using *only* those examples which pass through all the previous stages.

4. **Sequential Training: AdaBoost** Each stage of the classifier is trained using AdaBoost with decision stump as the base learner. This is essentially the well known Viola-Jones cascade [10]. The Viola-Jones cascade was developed in the context of face detection where we have a huge amount of very cheap features. There is no explicit notion of feature groups and different costs. We adapted the Viola-Jones cascade to our problem by training each stage of the cascade using only the features belonging to that group.

5. **Sequential Training: LDA** A sequentially trained classifier where each stage of the classifier is trained using linear discriminant analysis.

6. **Proposed soft cascade** $\beta > 0$ The proposed method of jointly training the cascade by taking into consideration the computational cost. We report results for $\beta = 10N, 100N$, and $1000N$ where N is the number of examples in the training set.

Choosing the thresholds: Once the classifier is trained we need to choose the threshold for each stage of the classifier. There has been a lot of work on choosing the thresholds for each stage in a conventional sequential cascade design [10, 1]. We choose our thresholds by doing an exhaustive two-level hierarchical grid search over a range of thresholds for each stage such that the area under the ROC curve for the training set is maximized. For the proposed method the entire cascade has to be trained jointly only once and the thresholds can be chosen as a post-processing step. For sequential training for each choice of the thresholds we have to retrain the cascade.

Evaluation Procedure: For the first two datasets we randomly select 70% of the data for training and 30% for testing. The split was done such that the ratio of the number of positive to negative examples was the same for both the sets. We report the resulting area under the ROC curve

Table 1: *Results for the Lung and Rectum Cancer datasets.* We randomly select 70% of the data for training and 30% for testing. The results shown are averaged over 10 such repetitions. The cost is normalized so that using all the available features has a cost of 1. For the proposed cascade we show results for $\beta = 10N, 100N$, and $1000N$ where N is the number of examples in the training set. The statistically significant values are marked as ✓ (as assessed by a two-sample t-test at 5% significance level against the Proposed soft cascade with $\beta = 0$ (marked $\star$)) .

	Training set		Testing set	
	AUC	Cost	AUC	Cost
	mean[± std]	mean[± std]	mean[± std]	mean[± std]
Lung Cancer				
(1) Single stage classifier	0.87[± 0.05]	✓ 1.00[± 0.00]	0.79[± 0.12]	✓ 1.00[± 0.00]
(2) Proposed soft cascade $\beta = 0$	$\star$ 0.84[± 0.06]	$\star$ 0.35[± 0.10]	$\star$ 0.72[± 0.11]	$\star$ 0.37[± 0.08]
(3) Sequential Training via Logistic Regression	✓ 0.78[± 0.05]	✓ 0.46[± 0.03]	0.71[± 0.09]	✓ 0.45[± 0.08]
(4) Sequential Training via AdaBoost	0.81[± 0.04]	✓ 0.51[± 0.02]	✓ 0.63[± 0.05]	✓ 0.68[± 0.08]
(5) Sequential Training via LDA	✓ 0.79[± 0.02]	✓ 0.55[± 0.03]	0.70[± 0.03]	✓ 0.66[± 0.08]
(6) Proposed soft cascade $\beta = 10N$	0.82[± 0.06]	0.32[± 0.06]	0.73[± 0.12]	0.35[± 0.12]
(7) Proposed soft cascade $\beta = 100N$	0.80[± 0.06]	0.30[± 0.05]	0.70[± 0.11]	0.35[± 0.11]
(8) Proposed soft cascade $\beta = 1000N$	0.80[± 0.06]	✓ 0.26[± 0.02]	0.70[± 0.11]	✓ 0.27[± 0.10]
Rectum Cancer				
(1) Single stage classifier	✓ 0.84[± 0.03]	✓ 1.00[± 0.00]	0.83[± 0.06]	✓ 1.00[± 0.00]
(2) Proposed soft cascade $\beta = 0$	$\star$ 0.80[± 0.04]	$\star$ 0.61[± 0.10]	$\star$ 0.79[± 0.06]	$\star$ 0.59[± 0.09]
(3) Sequential Training via Logistic Regression	✓ 0.75[± 0.02]	✓ 0.72[± 0.13]	0.76[± 0.09]	✓ 0.70[± 0.10]
(4) Sequential Training via AdaBoost	✓ 0.89[± 0.01]	✓ 0.70[± 0.08]	0.73[± 0.10]	✓ 0.68[± 0.09]
(5) Sequential Training via LDA	0.82[± 0.05]	0.68[± 0.14]	✓ 0.71[± 0.09]	0.63[± 0.12]
(6) Proposed soft cascade $\beta = 10N$	0.78[± 0.06]	0.58[± 0.08]	0.79[± 0.06]	0.57[± 0.08]
(7) Proposed soft cascade $\beta = 100N$	0.76[± 0.09]	✓ 0.53[± 0.06]	0.77[± 0.04]	✓ 0.50[± 0.07]
(8) Proposed soft cascade $\beta = 1000N$	✓ 0.73[± 0.08]	✓ 0.51[± 0.08]	0.76[± 0.08]	✓ 0.48[± 0.08]

(AUC) both for the training and the test set as our performance metric. We also report the average cost per patient. The cost is normalized so that using all the available features has a cost of 1. The results are averaged over 10 repetitions, and both the mean and standard deviation are reported. For the LungCAD dataset the classifier is training on the training set and we show the Free Reponse Receiver Operating Curve (FROC) for an independent test set. The FROC is a plot of the nodule level sensitivity vs the number of false positives per CT scan.

7.2 Results

Table 1 shows the results both on the training as well as the test set for the Lung cancer and the Rectum cancer datasets. For the proposed cascade we show results for varying values of β, which controls the emphasis we place on minimizing the cost over accuracy. Note that $\beta = 0$ corresponds to training the cascade without taking into consideration the computational cost. Larger the β more is the emphasis on reducing the cost. The statistically significant values–as assessed by a two-sample t-test against the proposed soft cascade with $\beta = 0$–are marked as ✓. We make the following observations:

1. As expected among all the methods the single stage classifier is the most accurate. This corresponds to the classifier trained by acquiring all the features. However it is the most expensive in terms of cost. The average cost per patient for this model is one. The cost is normalized so that using all the available features has a cost of one.

2. The proposed soft cascade with $\beta = 0$ has a lower performance (around $3 - 7\%$ lower) in terms of the AUC but is significantly cheaper (2-3 times) than the single stage classifier. Even if the cascade is trained with $\beta = 0$, at test time some data samples are rejected early (e.g. after the first or second stage) and therefore we do not obtain/compute the remaining features for them.

3. The proposed method of jointly training the cascade shows a superior performance (both in terms of the AUC and the cost) in comparison to the same classifier trained in a sequential manner (see lines (2) and (3) in the table). While the improvement of the AUC on the test set is not statistically significant we obtain quite significant improvements on the cost.

4. In terms of the cost the proposed method was superior than the sequentially trained adaboost or the LDA cascade. (see lines (2), (4), and (5) in the table)

5. For the proposed method increasing β reduces the cost and at the same time reduces the accuracy (though not that significantly). Only our method gives us a knob in terms of β to achieve our desired tradeoff between accuracy and cost.

Similar results can be observed for the LungCAD dataset. Figure 2 shows the FROC curves on an independent test

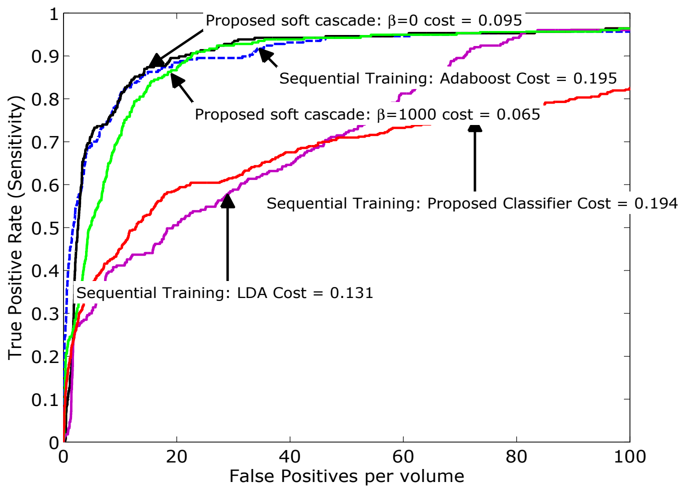


Figure 2: FROC Curves for the various cascade design methods on the LungCAD test set.

set. For this dataset the proposed joint training gives us a significant improvement over the same cascade trained sequentially. Also the proposed cascade is 10 times cheaper than a single stage classifier. Increasing β reduces the cost further with a slight drop in the performance.

8. CONCLUSIONS AND FUTURE WORK

We proposed a method to train a cascade of classifiers when groups of features are obtained together as a result of a sensing operation (*i.e.* entire group of features is obtained at the same cost). The classifier was trained jointly using the notion of soft cascades. We also demonstrated that by explicitly incorporating the computational cost into the algorithm we can achieve the desired tradeoff between accuracy and cost.

One assumption we have made is that prior to training the cascade the ordering of the different stages in the cascade is fixed. Typically the cascade is ordered by the feature acquisition time, *i.e.*, would like the first stage of our classifier to use the cheapest features and the most expensive features at the later stage. However this may not be the most optimal strategy in terms of both accuracy and cost. We are currently exploring strategies to formulate the automatic selection and ordering of different feature groups.

9. REFERENCES

[1] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 236–243, 2005.

[2] C. Capirci, L. Rampin, and et al. Sequential FDG-PET/CT reliably predicts response of locally advanced rectal cancer to neo-adjuvant chemo-radiation therapy. *Eur J Nucl Med Mol Imaging*, 34:1583–1593, 2007.

[3] C. Dehing-Oberije, D. De Ruysscher, and et al. Tumor volume combined with number of positive lymph node stations is a more important prognostic factor than TNM stage for survival of non-small-cell lung cancer patients treated with (chemo)radiotherapy. *Int J Radiat Oncol Biol Phys.*, 70(4):1039–1044, 2007.

[4] M. Dundar and J. Bi. Joint Optimization of Cascaded Classifiers for Computer Aided Detection. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[5] A. Genkin, D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[6] S. Ji and L. Carin. Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 40(5):1474–1485, 2007.

[7] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. An expected utility approach to active feature-value acquisition. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 745–748, 2005.

[8] R. B. Rao, J. Bi, G. Fung, M. Salganicoff, N. Obuchowski, and D. Naidich. LungCAD: a clinically approved, machine learning system for lung cancer detection. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1033–1037, 2007.

[9] V. S. Sheng and C. X. Ling. Partial example acquisition in cost-sensitive learning. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 638–646, 2007.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001.

[11] T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

[12] C. Zhang and P. Viola. Multiple-instance pruning for learning efficient cascade detectors. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1681–1688. MIT Press, Cambridge, MA, 2008.

[13] T. Zhang and F. Oles. Text Categorization Based on Regularized Linear Classification Methods. *Information Retrieval*, 4(1):5–31, 2001.

APPENDIX

A. DERIVATIVES FOR THE CYCLIC COORDINATE DESCENT OPTIMIZATION

A.1 Componentwise derivatives when $w_t \neq 0$

Because of the $\|\mathbf{w}\|_1$ term the derivatives are defined only when $w_t \neq 0$. The first derivative can be written as

$$J^{'}(w_t) \quad = \quad = -\sum_{i=1}^{N}\left[\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right]\frac{\partial p_i}{\partial w_t} + \alpha\ \mathrm{sgn}(w_t) + \beta\frac{\partial \mathcal{T}(\mathbf{w})}{\partial w_t}, \tag{20}$$

where $\mathrm{sgn}(z) = 1$ if $z > 0$ and -1 is $z < 1$. The second derivative is given by

$$J^{''}(w_t) \quad = \quad -\sum_{i=1}^{N}\left[\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right]\left(\frac{\partial^2 p_i}{\partial^2 w_t}\right) - \left[\frac{y_i}{p_i^2} + \frac{1-y_i}{(1-p_i)^2}\right]\left(\frac{\partial p_i}{\partial w_t}\right)^2 + \beta\frac{\partial^2 \mathcal{T}(\mathbf{w})}{\partial^2 w_t}. \tag{21}$$

Define $\mathbf{1}_{t\in\mathcal{C}_j} = 1$ if the feature x_t is used by the stage $\mathcal{C}_j$ and zero other wise. Then

$$\frac{\partial p_i}{\partial w_t} \quad = \quad p_i(\mathbf{x}_i)_t \sum_{j=1}^{K}(1-\sigma(f_{\mathcal{C}_j}(\mathbf{x}_i))\mathbf{1}_{t\in\mathcal{C}_j}. \tag{22}$$

$$\frac{\partial^2 p_i}{\partial^2 w_t} = \frac{1}{p_i}\left(\frac{\partial p_i}{\partial w_t}\right)^2 - p_i(\mathbf{x}_i)_t^2 \sum_{j=1}^{K}\sigma(f_{\mathcal{C}_j}(\mathbf{x}_i))(1-\sigma(f_{\mathcal{C}_j}(\mathbf{x}_i))\mathbf{1}_{t\in\mathcal{C}_j}. \tag{23}$$

$$\frac{\partial \mathcal{T}(\mathbf{w})}{\partial w_t} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)_t \sum_{j=2}^{K} t_j \left[\prod_{l=1}^{j-1}\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right)\right]\sum_{l=1}^{j-1}(1-\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right))\mathbf{1}_{t\in\mathcal{C}_l}. \tag{24}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{T}(\mathbf{w})}{\partial^2 w_t} \quad &= \quad \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)_t^2 \sum_{j=2}^{K} t_j \left[\prod_{l=1}^{j-1}\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right)\right]\left[\sum_{l=1}^{j-1}(1-\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right))\mathbf{1}_{t\in\mathcal{C}_l}\right]^2 \\ &- \quad \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)_t \sum_{j=2}^{K} t_j \left[\prod_{l=1}^{j-1}\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right)\right]\left[\sum_{l=1}^{j-1}\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right)(1-\sigma\left(f_{\mathcal{C}_l}(\mathbf{x}_i)\right))\mathbf{1}_{t\in\mathcal{C}_l}\right].\end{aligned} \tag{25}$$

A.2 Componentwise derivatives when $w_t = 0$

Since the l_1-penalty is not differentiable at $w_t = 0$ we write the directional derivative along the forward and the backward direction. Let $\mathbf{e}_t$ be the co-ordinate direction along which w_t varies. Then the directional derivatives are given by

$$J_{+}^{'}(w_t) = \lim_{\delta\to 0}\frac{J(\mathbf{w}+\delta\mathbf{e}_t) - J(\mathbf{w})}{\delta} = -\frac{\partial l(\mathbf{w})}{\partial w_t} + \alpha + \beta\frac{\partial \mathcal{T}(\mathbf{w})}{\partial w_t}, \tag{26}$$

and

$$J_{-}^{'}(w_t) = \lim_{\delta\to 0}\frac{J(\mathbf{w}-\delta\mathbf{e}_t) - J(\mathbf{w})}{\delta} = -\frac{\partial l(\mathbf{w})}{\partial w_t} - \alpha + \beta\frac{\partial \mathcal{T}(\mathbf{w})}{\partial w_t}. \tag{27}$$

The algorithm evaluates both $J_{+}^{'}(w_t)$ and $J_{-}^{'}(w_t)$. We attempt to update in both directions and see if either succeeds as suggested in [5].

A.3 Convergence Criterion

We declare convergence when $\frac{\sum_{i=1}^{n}|\Delta p_i|}{\sum_{i=1}^{n}|p_i|} \leq \epsilon$, where $|\Delta p_i|$ is the change in p_i between the beginning and the end of a cycle.