Using Classifier Cascades for Scalable E-Mail Classification

Jay Pujara
Dept. of Computer Science
University of Maryland
College Park, MD, USA 20742
jay@cs.umd.edu

Hal Daumé III
Dept. of Computer Science
University of Maryland
College Park, MD, USA 20742
hal@umiacs.umd.edu

Lise Getoor
Dept. of Computer Science
University of Maryland
College Park, MD, USA 20742
getoor@cs.umd.edu

ABSTRACT

In many real-world scenarios, we must make judgments in the presence of computational constraints. One common computational constraint arises when the features used to make a judgment each have differing acquisition costs, but there is a fixed total budget for a set of judgments. Particularly when there are a large number of classifications that must be made in a real-time, an intelligent strategy for optimizing accuracy versus computational costs is essential. E-mail classification is an area where accurate and timely results require such a trade-off. We identify two scenarios where intelligent feature acquisition can improve classifier performance. In **granular classification** we seek to classify e-mails with increasingly specific labels structured in a hierarchy, where each level of the hierarchy requires a different trade-off between cost and accuracy. In **load-sensitive classification**, we classify a set of instances within an arbitrary total budget for acquiring features. Our method, *Adaptive Classifier Cascades* (ACC), designs a policy to combine a series of base classifiers with increasing computational costs given a desired trade-off between cost and accuracy. Using this method, we learn a relationship between feature costs and label hierarchies, for granular classification and cost budgets, for load-sensitive classification. We evaluate our method on real-world e-mail datasets with realistic estimates of feature acquisition cost, and we demonstrate superior results when compared to baseline classifiers that do not have a granular, cost-sensitive feature acquisition policy.

1. INTRODUCTION

Electronic mail has become an integral part of daily communication, filling needs ranging from the delivery of financial statements to personal correspondence. The huge volume, and multifarious uses of email have an impact on users, who must dedicate time to organize and categorize the influx of messages. The popularity of the medium also requires service providers to operate at extremely large scale, handling billions of messages, totaling terabytes of data, each day.

The immense scale of e-mail provides over-sized computational challenges. For example, the vast majority of attempts to send e-mail consist of unwanted marketing messages ("spam"), which providers must reject or place in a separate folder. Moreover, webmail systems are increasingly attempting to use more involved processing in an effort to improve user experience. Examples of this functionality include detecting meeting invitations, creating slideshows from attached photographs, annotating metadata from social network profiles, recognizing shipment tracking information, and predicting a message's relevancy to a user. For each such application, a series of specialized and potentially expensive processing steps are necessary. Applying each set of steps to every message is not scalable or beneficial; meeting invitations generally do not contain shipment tracking information. To apply only relevant features to each message, providers must understand more about the type of message being sent, for example meeting invitations.

In these scenarios, the goal of a mail system is to classify a message with a set of increasingly specific labels, each with their own computational constraints, a task we refer to as **granular classification**. The categories occur at differing scales with a defined structure: at a coarse scale messages are considered undesirable ("spam") or desirable ("ham"), while desirable messages can be further classified at a finer scale as "business communication" or "social network notifications." Messages containing "business communication" may possibly be "shipment notifications" and as a result are candidates for extracting package tracking information. We refer to this set of labels and the relationship between them as a *label hierarchy*.

In the granular classification setting, the prediction of each label in the series may require a different trade-off between cost and accuracy, necessitating a different feature acquisition strategy. Classifying messages as ham or spam occurs at large-scale and minimizing the computational cost of classifying each message can be very important. Conversely, far fewer messages can be considered business communication, and accuracy may be paramount when deciding whether a certain message is a receipt for a received payment or a bill for past due expenses. We refer to the trade-off between cost and accuracy for a given prediction within the hierarchy as the *cost sensitivity* of that classification. This cost sensitivity may change depending on conditions, as our next problem scenario demonstrates.

While mail systems are providing a more powerful set of functions, a crucial requirement is high availability. Prob-

CEAS '11 September 1-2, 2011, Perth, Western Australia, Australia

Command state	Associated features
<Connect>	$\phi_1(\mathbf{x}) \rightarrow$ IP address of remote host
HELO	Identity of remote host
MAIL FROM	$\phi_2(\mathbf{x}) \rightarrow$ Sender of message
RCPT TO	Recipient of message
DATA	$\phi_3(\mathbf{x}) \rightarrow$ Message incl. headers

Table 1: SMTP commands and associated features

lems such as interruptions in network connectivity, machine failures, and attacks from malicious entities offer the potential to degrade mail service and impact broader system stability. Given such risks, conventional classification techniques are ill-suited, because they rely on static feature vectors that are the same for each message, potentially requiring the full text of each message before classification begins. We term this problem scenario **load-sensitive classification**, where the classifier uses information about system load, in the form of a classification budget, to intelligently choose a feature set for each message while maintaining high throughput. In this setting, during high load conditions, a mail system adapts to system load by decreasing the average feature cost for each instance and potentially suffering a reduction in classification accuracy.

Fortunately, e-mail messages also adhere to a protocol that allows features to be acquired incrementally, at differing costs. During an SMTP conversation, IP information arrives first, followed by sender and recipient information, the mail header, and finally the message content, as illustrated in Table 1. In this environment, successive features take increasingly longer to acquire and can have increasingly diverse values making acquisition expensive. These aspects of the feature costs provide the opportunity to build classifiers that exploit the feature structure and relative costs to help ensure system stability. During high load conditions, classifiers can favor cheaper features to increase the throughput of decisions. In the setting of a label hierarchy, cheap features can be used to make coarse judgments and progressively expensive features can be used for classification at a finer level. Furthermore, the structure imposed by feature costs and dependencies provides a compelling parallel to the structure of message categories. Relating these two structures provides the promise of making coarse-to-fine category judgments (Figure 1). By using cost-sensitive methods that emphasize incremental acquisition of features on the basis of acquisition cost, the ability to make granular classifications becomes a more computationally tractable problem.

A popular approach used to manage classification cost is the use of classifier cascades[12]. In such a setting, each instance is supplied to a series of classifiers that can each either output a result or pass the instance to the next classifier for further consideration. Generally, the initial classifiers have lower costs and see many instances while the final classifiers have higher costs and higher accuracy but see few instances. Classifier cascades have been successfully applied to a number of problems, such as generating optimal features for face detection[12] or reducing the number of tests in medical treatment[9]. A powerful aspect of this approach is the trade-off between cost and accuracy; allowing initial stages to classify more instances can reduce costs, but often with a reduction in accuracy.

Our contribution is to use a variant of classifier cascades,

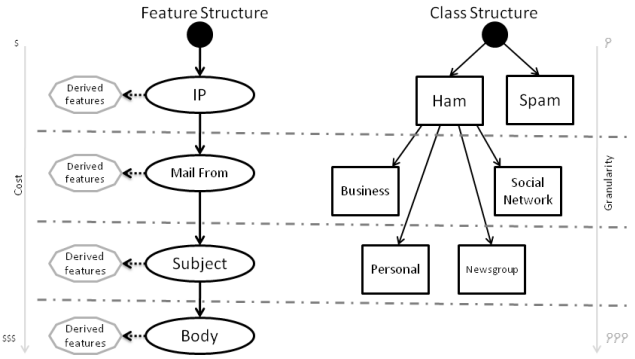


Figure 1: Relationship of feature cost and class granularity

Adaptive Classifier Cascades (ACC), to classify e-mail in two settings, minimizing a loss that is a combination of computational cost and classifier error. In the first setting, granular classification, the classifier makes a series of predictions from a label hierarchy where the relative contribution of computational cost varies at each level of the hierarchy. Our contribution is a method of learning cascade parameters that minimize this loss at each level of the hierarchy. In the second setting, the classifier cascade receives a classification budget as input and produces predictions where the average cost of classifying an instance does not exceed the budget. Our contribution is learning cascade parameters that allow the classifier to produce results given arbitrary cost budgets.

2. RELATED WORK

Computational cost of classification has been reduced through cascade classifiers following the approach of Viola and Jones [12]: instances are initially processed through computationally efficient classifiers for a coarse judgment and those instances judged interesting during the coarse classification are reconsidered with increasingly computationally demanding models. Recent work has sought a joint optimization of the different classifier stages by minimizing a Lagrangian combining cost and classification accuracy[10]. Cascade architectures for solving granular, coarse-to-fine problems are a topic of active research, and an early version of this work has been presented in such forums[1]. The problem of hierarchical text classification as initially presented in [6] supports feature sets tailored to the classification task at a particular level. Subsequent work has considered hierarchical classification of e-mail[5]. The question of actively acquiring features to improve classifier performance[11] both during training [7] and at test time [2] has also been studied.

3. PROBLEM SETTING

We introduce the problem of cost-sensitive supervised classification and the use of cascades as a tool for combining classifiers to solve cost sensitive problems in Section 3.1. We introduce two novel problems, granular classification and load-sensitive classification and provide a formal definition of these problems in Section 3.2.

3.1 Cost-Sensitive Classifier Cascades

In a basic supervised classification setting, a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$ is provided, containing instances

$\mathbf{x}_i \in \mathbf{X}$ and class labels $y_i \in \mathbf{Y} = \{-1, 1\}$. Each instance $\mathbf{x}$ is described by m sets of features,

$$\mathbf{x} = \langle \phi_1(\mathbf{x}) \dots \phi_m(\mathbf{x}) \rangle$$

The goal is to learn a mapping from the instances $\mathbf{X}$ to classes $\mathbf{Y}$, $h : \mathbf{X} \rightarrow \mathbf{Y}$. This is achieved by minimizing a loss function, $L(y, h(\mathbf{x}))$ over the training data.

$$\min_h \sum_{(\mathbf{x},y)\in\mathcal{D}} L(y, h(\mathbf{x}))$$

In *cost-sensitive classification*, the problem requires minimizing a loss function that includes a component proportional to the cost of classification, $C(h(\mathbf{x}))$. In our work, we ascribe this cost to feature acquisition or the computational overhead of computing a decision function. We designate a loss function sensitive to the cost $L_c(y, h(\mathbf{x}); \lambda)$, and refer to the parameter λ, which governs the contribution of cost to the loss function, as the *cost-sensitivity* of the loss function.

$$L_c(y, h(\mathbf{x}); \lambda) = L(y, h(\mathbf{x})) + \lambda C(h(\mathbf{x}))$$

$$\min_h \sum_{(\mathbf{x},y)\in\mathcal{D}} L_c(y, h(\mathbf{x}))$$

To learn the mapping from instances to labels, we employ a series of base classifiers, continuous-valued predictors, $f_1 \dots f_m$ that approximate h using a subset of the feature space. We define the i^{th} base classifier f_i to make predictions using the first i feature subsets, $\phi_1(\mathbf{x}) \dots \phi_i(\mathbf{x})$:

$$f_i : \langle \phi_1(\mathbf{x}) \dots \phi_i(\mathbf{x}) \rangle \rightarrow \mathbb{R}, 1 \leq i \leq m$$

The output of each of these base classifiers is the decision margin, or the distance of the instance from the decision boundary. Values close to 0 indicate uncertainty about the prediction. Each base classifier has an associated cost or complexity, $c_1 \dots c_m$, attributed to the cost of acquiring the subset of features used by the classifier. In this setting, since base classifiers progressively incorporate more features, the costs are strictly ordered, such that $c_1 < c_2 \dots < c_m$.

Classifier cascades provide a framework for combining the base classifiers to minimize a cost-sensitive loss, providing the required trade-off between classification accuracy and computational cost. We can formulate the learned hypothesis $h(\mathbf{x})$ as a classifier cascade $\mathcal{F}(\mathbf{x})$ given base classifiers $f_1 \dots f_m$ and with parameters $\gamma_1 \dots \gamma_m$ denoted as $\mathcal{F}(f_1 \dots f_m; \gamma_1 \dots \gamma_m)(\mathbf{x})$. As shorthand, we introduce the notation $\mathcal{F}_i(\mathbf{x})$ for the cascade $\mathcal{F}(f_1 \dots f_i; \gamma_1 \dots \gamma_m)(\mathbf{x})$. We set $\mathcal{F}_m(\mathbf{x}) = f_m(\mathbf{x})$. For $i < m$, we define

$$\mathcal{F}_i(\mathbf{x}) = \mathcal{F}(f_1 \dots f_i; \gamma_1 \dots \gamma_m)(\mathbf{x}) = \begin{cases} f_i(\mathbf{x}) & \text{if } |f_i(x)| \geq \gamma_i \\ \mathcal{F}_{i+1}(\mathbf{x}) & \text{otherwise} \end{cases}$$

The computational cost of the cascade, $\mathcal{C}(\mathbf{x})$, can be expressed as a similar recurrence:

$$\mathcal{C}_i(\mathbf{x}) = \mathcal{C}(f_1 \dots f_i; \gamma_1 \dots \gamma_m)(\mathbf{x}) = \begin{cases} c_i & \text{if } |f_i(x)| \geq \gamma_i \\ \mathcal{C}_{i+1}(\mathbf{x}) & \text{otherwise} \end{cases}$$

Note that this definition of the cascade is different from the conventional treatment, where a common assumption is that negative instances vastly outnumber positive instances, and only positive instances are considered by subsequent stages. Instead, we borrow from boosting the idea of providing additional consideration to those instances that are likely to be poorly classified by our existing classifier.

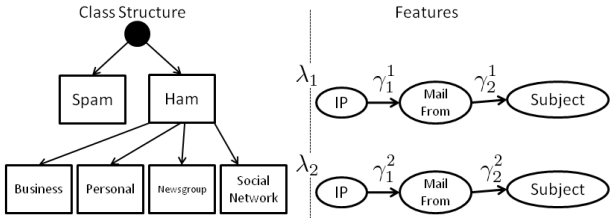


Figure 2: Parameters learned for a granular classification task with a 2-level hierarchy

Since the base classifiers produce a continuous result, their output can be considered a confidence or decision margin. Consequently, we are able to parameterize the cascade with $\gamma_1 \dots \gamma_m$. We refer to the parameters $\gamma_1 \dots \gamma_m$ as a *policy*. The cascade classifier learns a policy by choosing parameters that control the circumstances where an instance is evaluated by a subsequent stage. The choice of the policy provides a method to minimize the cost-sensitive loss of the classifier.

3.2 Extensions to Cost-Sensitive Cascades

The most basic goal in this problem setting is to minimize a loss function sensitive to classification cost, $L_c(y, h(\mathbf{x}); \lambda)$. In our formulation of classifier cascades, if the base classifiers are fixed and λ is given, this can be accomplished by learning the parameters $\gamma_1 \dots \gamma_m$ that minimize this objective. However, we consider two scenarios, granular classification and load-sensitive classification, which provide additional challenges to the classifier cascade model.

Granular Classification

In the e-mail domain it is necessary to make multiple, coarse-to-fine judgments, a problem of **granular learning**: for a series of related tasks, each task has differing cost-sensitivity. This is directly applicable when predicting a series of class labels in a hierarchy, with differing computational constraints for each prediction. Coarse predictions, such as distinguishing between "ham" and "spam" may have higher cost-sensitivity than fine-grained predictions such as deciding between categories such as "newsletter" and "social network." Due to this pattern of cost-sensitivity, the coarse classification may be limited to using few features to limit feature acquisition cost, while the fine classification task is able to make predictions using a larger feature set. Features acquired for a coarse judgment, such as the sending behavior of a domain, can be reused in the fine classification task with little or no overhead. However mistakes made in the coarse judgment can cause errors to propagate to the fine judgment, as the classifier undertakes different prediction tasks at finer levels based on previous decisions. System designers can influence the performance of a system across diverse operating profiles in terms of cost and accuracy of coarse and fine tasks by appropriately choosing the cost-sensitivity.

While predictions in the granular classification setting occur on the same set of instances and have related labels, the cost-sensitivity parameter, λ is different at each level of the hierarchy. Our goal in this setting is to learn a policy by

choosing parameters $\langle \gamma_1 \dots \gamma_m \rangle$ that minimize cost-sensitive loss at each level of the hierarchy. Specifically, for a classifier cascade, $h_l(\mathbf{x})$, at each level l of the hierarchy we must learn parameters $\langle \gamma_1^l \dots \gamma_m^l \rangle_l$, minimizing a loss with cost-sensitivity, λ_l. This task is illustrated in Figure 2.

Load-Sensitive Classification

Another circumstance commonly encountered in the e-mail domain is performing classification while faced with a computational budget. Given a computational budget, B, governing the average cost of classifying an instance we may wish to choose a policy that adheres to an additional constraint that $\mathcal{C}(\mathbf{x}) < B$. However, this computational budget may not be known in advance, and can depend on prevailing conditions. Users may send more e-mail at specific times of day or during significant events, servers may suffer outages, or malicious users may attack the system. Each of these circumstances might require a different computational budget to maintain the throughput of the classification system.

Since the budget necessary for classification is not known in advance, the problem of **load-sensitive classification** requires learning a policy that can meet an arbitrary budget constraint. In essence, we must learn a function $Z(B, h(\mathbf{x})) \rightarrow \langle \gamma_1 \dots \gamma_m \rangle$, such that Z chooses γ parameters for an arbitrary budget B given a known hypothesis $h(\mathbf{x})$.

As an example, in a denial-of-service (DoS) attack, thousands of machines may simultaneously attempt to access a small set of servers, causing an influx of requests. While a decision system might normally use a robust set of features, during the DoS attack a system administrator might choose a lower budget for classification. Using a load-sensitive classification system, parameters are chosen that decrease the average cost of classification by restricting the features used for many decisions. This allows the system to maintain connectivity for legitimate traffic. Using fewer features may reduce accuracy, a tradeoff explored in Section 5.

4. ADAPTIVE CASCADE CLASSIFIERS

In this section we present the method for minimizing cost-sensitive loss we refer to as the *Adaptive Cascade Classifier*. Using the formulation for classifier cascades introduced in Section 3, we detail a method to learn policies that minimize cost-sensitive loss and extend this method to solve the tasks of granular classification and load-sensitive classification.

Learning policies for different load conditions or classification granularity is a difficult task. One obstacle is that the objective function $_c(y, h(\mathbf{x}))$ is not continuously differentiable with respect to $\gamma_1 \dots \gamma_m$; changing one γ value can shift a number of instances from one classifier stage to the next. Since the classification accuracy of a subsequent stage may vary, the objective function is non-monotonic, posing challenges for approaches that find local optima.

These aspects of the problem require a global optimization in the combinatorial parameter space. To perform this global optimization, we use grid search to learn the parameters that minimize the objective function, as detailed in Algorithm 1. While this approach is not generally tractable, it is appropriate for the domain of e-mail classification where training instances are many orders of magnitude smaller than classification requests, and few constraints exist on the classifier training time or resources.

4.1 Granular Classification

Algorithm 1 ACC-GRID-SEARCH: Find optimal cost-sensitive policy

Require: Dataset $\mathcal{D}$
Require: Cascade Classifier $\mathcal{F}$
Require: Cost-Sensitivity parameter λ
Require: GEN-COMB, a function sampling combinations of values from the range of $\gamma_1 \dots \gamma_m$
  MIN $\leftarrow \infty$
  for all $\langle \gamma_1 \dots \gamma_m \rangle$ = GEN-COMB **do**
    LOSS $\leftarrow 0$
    for all $(\mathbf{x}, y) \in \mathcal{D}$ **do**
      LOSS $\leftarrow$ LOSS $+ L_c(y, \mathcal{F}(\mathbf{x}); \lambda)$
    end for
    if LOSS $<$ MIN **then**
      MIN $\leftarrow$ LOSS
      MINPARAMS $\leftarrow \langle \gamma_1 \dots \gamma_m \rangle$
    end if
  end for
  return MINPARAMS

The problem of finding optimal parameters is magnified when learning related tasks, as the parameter choices made to find the optimal policy for a coarse task, at the top of the label hierarchy, can change the distribution of instances presented to a fine task at a more granular level in the hierarchy. Performing a joint optimization of cost-sensitive loss requires weighing the contribution of the loss function at each stage, and adjusting γ parameters at multiple levels in the hierarchy to minimize this loss. In this work, we approximate this joint optimization by sequentially finding a global optimum for each level of the hierarchy using ACC-GRID-SEARCH, as shown in Algorithm 1. Using this method, we can ensure that each classifier in the hierarchy optimizes over the appropriate subset of instances from previous levels.

4.2 Load-Sensitive Classification

In a load-sensitive setting, the optimization problem described earlier also becomes more complex. Since the goal is to provide parameters for arbitrary cost budgets, we must solve the meta-learning problem of discovering the mapping from $\mathbb{R} \rightarrow \langle \gamma_1 \dots \gamma_m \rangle$. A grid search can provide cost information for many parameter values, allowing us to create a mapping from a subset of possible budgets to a set of parameters.

One issue of using cost information from grid search is the possibility of overfitting. While a policy may have an acceptable cost on the training data, a small difference in the distribution of the test instances could cause the classifier to exceed the cost budget. Features missing in the training dataset could result in low margins and high classification costs. In some cases, the cost budget is absolute, and exceeding the budget is not possible. To increase the robustness of the load-sensitive classification we add a regularization term.

The regularization term is generated by measuring variance in the cost predictions made by a specific policy and adjusting the estimated cost for prediction. This is achieved by modifying the load-sensitive constraint: $\mathcal{C}(\mathbf{x}) + \Delta\sigma < B$, where σ is the standard deviation of classification cost and Δ is a constant tuned based on the flexibility of the budget.

Class	Count
Spam	531
Business	187
Social Network	233
Newsletter	174
Personal/Other	102

Table 2: Message Categories, Counts in Yahoo! Mail Dataset

5. EXPERIMENTAL EVALUATION

In this section we discuss the evaluation of Adaptive Classifier Cascades on a real-world dataset from Yahoo! Mail users and the TREC-2007 Spam Corpus. We begin by introducing relevant aspects of the e-mail domain, provide details about the datasets we use and the calculations performed for feature costs, and end by defining the specific tasks and baselines used for evaluation and the associated results.

5.1 E-Mail Domain

E-mail is generally delivered to Mail Transfer Agents (MTAs) using the Simple Mail Transfer Protocol (SMTP)[8]. This protocol defines a conversation consisting of a well-ordered set of commands (Table 1). These commands correspond to the ordered feature sets, $\phi_1(\mathbf{x}) \dots \phi_m(\mathbf{x})$ considered in the problem description. After each command the MTA must send a response code, indicating whether the command was successful or resulted in an error, and update its internal state.

The first piece of information available to the MTA is the IP address of the remote sender, which arrives as soon as the sender connects. As the conversation continues, the sender will send the "MAIL FROM" command and provide an e-mail address. The sender must provide one or more recipients using the "RCPT TO" command. Finally, the sender will enter the "DATA" state and send the message. Usually the message will consist of headers and content. The headers contain metadata about the message, including routing information as well as items such as the date and time the message was sent, the sender's name, and the subject.

The structure of this conversation lends itself to coarse-to-fine processing. The IP address can differentiate between known hosts with a history of sending messages of a specific type and unknown hosts. In particular, known senders of spam can be given an error after the first command (e.g., after acquiring the first level features), effectively blocking the sender.

5.2 Dataset

We evaluated on data sampled from a month-long time window of e-mail data exchanged by users of Yahoo! Mail.[1] To create a tractable experiment, the feature set was limited to four types of features: remote IP address, sender mailfrom domain, sender mailfrom address, and tokens from the message subject. We representatively sampled approximately 100 feature values from each of the four feature types, yielding 432 features. We then randomly sampled messages from the same month of e-mail data, restricting the sample to messages containing at least one of the selected features values. For each of the 432 features, up to three messages were randomly selected, yielding a total 1227 messages. The full feature vectors, encompassing all possible IP, sender and subject features found in the 1227 training messages, were constructed for these messages.

Sampled messages were categorized by a human expert into five categories: "business", "social network", "newsletter", "personal/unknown" and "spam". The frequency of each category is shown in Table 2.

In order to demonstrate the reproducibility of our results, we also use the TREC-2007 Public Spam Corpus[3], a standard mail dataset used in many studies, to evaluate the load-sensitive classification task. We prepared the dataset by using the final "Received" header for IP information, using the initial "From" line for sender features, and extracting the "Subject" header for the subject features. Messages were removed when a non-private IP did not occur in the final Received header, From information was not available, or when the Subject consisted of non-ASCII characters. This procedure yielded 47194 total messages, 39055 spam and 8139 ham.

[1]This data was acquired when the first author was employed by Yahoo! Mail. With the permission and support of Yahoo! through their Academic Relations department, we have been able to continue to use an anonymized version of this dataset for research purposes.

Feature Set	C_n	C_s	C
IP$\rightarrow \phi_1(\mathbf{x})$	0	5.035	.168
MailFrom$\rightarrow \phi_2(\mathbf{x})$	3	6.640	.322
Subject$\rightarrow \phi_3(\mathbf{x})$	8	7.299	.510

Table 3: Feature Classes and Costs

5.3 Feature Classes and Costs

We defined three feature sets, IP features (ϕ_1), sender features (ϕ_2), and subject features (ϕ_3). Each of these feature sets was attributed a cost, C (Table 3). Costs were normalized to their fractional share of the cost of the entire feature vector, so that acquiring all features for a message corresponds to incurring a cost of 1.

Two factors were considered when computing the cost of a feature: the network traffic necessary to acquire the feature (C_n) and the storage required to hold all possible feature values (C_s). In practice these costs can be weighted by their computational impact; here we weighted them equally. The normalized combination of these two costs is shown as C. Since MTAs communicate with senders via TCP, the network cost can be quantified in packets exchanged with the sender. The storage cost of features was computed by calculating the entropy of feature values in the sampled messages. The entropy represents the number of bits needed to optimally encode the feature values, and this is the best-case scenario for storing feature information.

A notable observation about the feature costs in mail classification is that they increase by an order of magnitude at each stage of the SMTP conversation. The estimated cost of the MailFrom features is almost double that of the IP features, and the subject features are similarly expensive relative to the MailFrom features. This structure of feature costs motivates cost-sensitive approaches to the problem.

5.4 Classification Tasks

We explore two classification tasks where features are actively acquired to minimize a combination of cost and classifier error, governed by the constant λ. Base classifiers are

trained using MegaM[4] at the IP, sender and subject feature levels. The loss function used is 0-1 loss, resulting in a value of 1 for each incorrectly classified instance. In each case, the classifier learns two parameters, γ_1 and γ_2, controlling when sender information or header information, respectively, are acquired on the basis of the classifier margin. Experiments are done using 10-fold cross validation, where each method is trained using 90% of the data and evaluated on the remaining 10%. All results show are averages across 10 folds unless otherwise noted. Results are compared with a baseline classifier, and paired t-tests at the ($p < .05$) level are used to assess significance. Significance tests were only performed for overall error and overall cost.

Granular Classification

In the granular classification task, we divide the problem into two stages. The first, a "coarse" stage, involves determining if a message is "spam" or non-spam ("ham"). The tradeoff between cost and accuracy for this stage is controlled by the cost-sensitivity parameter, λ_c, and the classifier learns parameters γ_{c1}, γ_{c2} that minimize the cost-sensitive loss for the coarse task. We separately train a second, "fine" stage using only ham messages, where the goal is to classify the message into one of four granular categories ("business", "newsletter", "personal", "social network"). In the multiclass setting, we use the ratio of probabilities of the second most likely class and the most likely class for the classification margin. Here the cost-accuracy trade-off is controlled by λ_f, learning parameters γ_{f1}, γ_{f2}.

When classifying instances, we first apply the coarse classifier, and actively acquire features based on the learned γ_c parameters to decide whether messages are ham or spam. All messages judged to be ham by the coarse stage are further classified by the fine stage, using the already acquired features and acquiring additional features as dictated by the γ_f parameters. Loss is reported with respect to the number of incorrectly classified instances across both tasks.

We compare the results of our approach with a naive baseline that uses a static feature acquisition strategy. In this baseline, a fixed feature vector is used to perform classification such that the cost of prediction for each instance is equal, since the same set of features are used for all instances. We compare against three feature vectors including progressively more features: $\phi_1(\mathbf{x})$, $\langle \phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \rangle$ and $\langle \phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \phi_3(\mathbf{x}) \rangle$.

Load-Sensitive Classification

For the load-sensitive classification task, the goal is to maximize classifier accuracy while performing classification under a computational cost budget. Since computational costs have been normalized from 0 to 1, the classifier maps each cost in this range (at a granularity of 10^{-3}) to the optimal γ_1, γ_2 parameters.

To increase robustness, the standard deviation in classification cost, σ weighted by constant Δ is used as part of the cost estimate. This avoids overfitting cost estimates in the training data when learning a policy and models the diversity of instances at prediction time.

We test the classifier in the same $(0, 1)$ range of budgets, measuring the classification accuracy and the actual classification cost at each budget level. The metrics used for evaluation include the loss, the percentage of instances where classification cost exceeds the budget, and the average excess computational cost when exceeding the budget.

The baseline we compare against is a naive approach where the largest fixed feature vector with cost below the budget is used to classify each instance. In this setting, each instance will have the same classification cost for each budget, but this classification cost can increase as the budget increases. By virtue of this design, the naive classifier will never exceed the cost budget.

5.5 Results

The goal of this work is to demonstrate techniques that deal with the common trade off between the accuracy of predictions and the cost of classification. In Section 4 we introduced methods to train a cascade classifier model to minimize a combination of classification error and classification cost. In the results, we focus on how using this method can allow the design of classifiers meeting diverse needs.

Granular Classification Results

One scenario we consider is granular classification, where there is a relationship among classification tasks, and parameters control the cost-sensitivity of these related tasks. Results for the granular classification task are presented in Table 4, with bold values indicating significant values based on paired t-tests at a significance level of ($p<.05$).

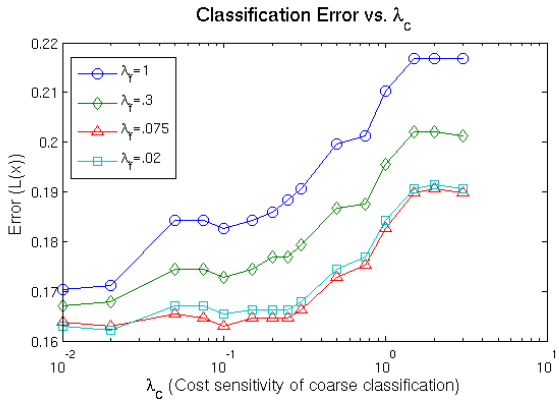
Cascade classifiers offer significant improvements over the baseline performance. When cost is very important, as in the first pair of rows, it is necessary to operate with the bare minimum of features. In this case, there is no way to reduce the cost beyond the baseline which acquires a single feature. However, a cascade classifier parameterized with $\lambda_c = 1.5, \lambda_f = 1$ can reduce error even with small increases in the classification costs.

In the second pair of rows, the baseline classifier acquires two features for each instance, while the cascade classifier parameterized with $\lambda_c = .1, \lambda_f = .075$ adaptively acquires features. The learned model allows the cascade classifier to outperform the baseline in both cost and accuracy.

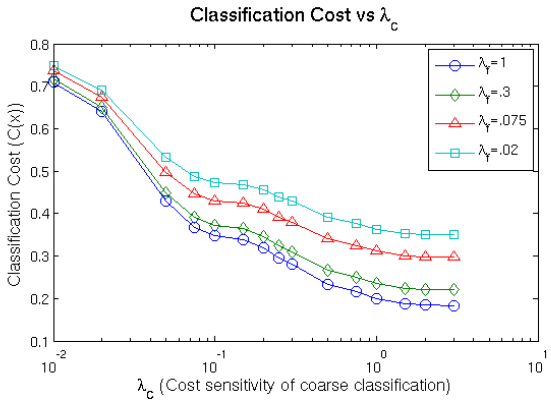
Finally, the third pair of rows shows a baseline classifier using the full feature set and a cascade classifier parameterized with $\lambda_c = .02, \lambda_f = .02$. In contrast to the first comparison, where no fewer features could be acquired, in this setting no more features can be acquired. Consequentially, the overall error of both classifiers is equal. However, the cascade classifier can achieve this accuracy at significantly lower costs by acquiring features adaptively.

As the range of cost profiles in the results suggests, choosing the parameters λ_c and λ_f correctly is crucially important to the classification outcome. Moreover, due to the relationship between tasks in the granular classification setting, the choice of one parameter can have an impact on the potential influence of other parameters. This is evidenced in Figure 3(a) where the effect of λ_c on classification error is shown for differing values of λ_f. When λ_c is low, the coarse classification stage aggressively acquires features and makes the cost sensitivity of future tasks immaterial as most features have already been acquired. As the cost sensitivity of the coarse task increases, the choice of λ_f has greater influence on the overall error.

Aggressive feature acquisition produce lower error rates, but can result in a corresponding increase in cost, as seen in Figure 3(b). As seen with classification error, when λ_c is low, the choice of λ_f has little impact. However, at higher

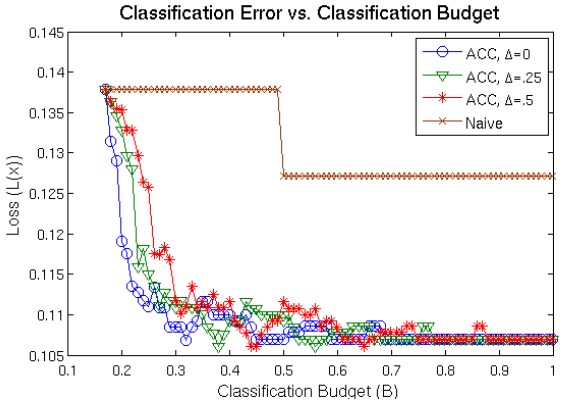


(a) Classification error of classifier cascades with different cost-sensitivities for coarse and fine tasks

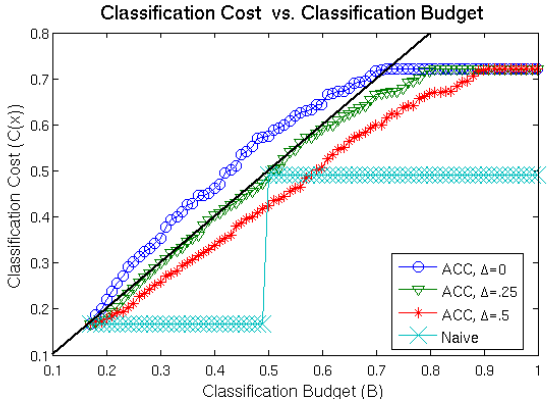


(b) Classification cost of classifier cascades with different cost-sensitivities for coarse and fine tasks

Figure 3: Relationship of (a) error and (b) cost to the cost-sensitivity parameters of a classifier cascade in a granular classification task in the Yahoo! Mail dataset



(a) Classification error vs. budget constraints, with different values of regularization parameter Δ



(b) Classification cost vs. budget, with different values of regularization parameter Δ. The bold line indicates cost equal to the budget

Figure 4: Relationship of (a) error and (b) cost to the classification budget in a load-sensitive classification setting for the Yahoo! Mail dataset

levels of coarse cost sensitivity the classifiers that produce predictions with lower error also incur greater costs.

Load-Sensitive Classification Results

In the second task of interest, load-sensitive classification, the goal is to perform classification within a specified computational budget. This objective is complicated by both overfitting the budget and variability in the budget predictions, so regularizing the estimates based on their standard deviation is necessary. We present the results of experiments in Table 5 for the Yahoo! Mail dataset and Table 6 for the TREC-2007 spam corpus. These results include trends for error and adherence to the budget over the range of all possible budgets (.168 to 1) in increments of .001. We also investigate how the value of the regularization constant Δ impacts the accuracy of the predicted budget.

While the load sensitive cascades show lower error on the prediction task, the regularization determines whether the classification occurs within the classification budget. When no regularization is used, the load-sensitive cascade exceeds the budget on test data for the majority of budgets in the Yahoo! Mail dataset and for almost 40% of the budgets in the TREC dataset, with average excesses of .054 and .059 respectively. In the Yahoo! Mail dataset, when the regularization constant is .25, budget excesses are both small and rarer, and with a regularization constant of .5, the classifier almost never exceeds the budget. For the TREC dataset, a regularization constant of .25 results in budget predictions that never exceed the desired budget on test instances.

Figure 4(a) shows that the error rate for all cascade classifiers converges relatively quickly as the budget increases, regardless of the Δ parameter. The cost of classification at test time relative to the budget is shown in Figure 4(b), and clearly demonstrates the benefits of regularization. The observed costs at $\Delta = 0$ usually overshoot the budget, while the costs at $\Delta = .5$ are consistently below the allowed budget. At $\Delta = .25$, the predicted costs follow the budget very closely. The naive method can only take full advantage of the budget when the budget matches feature costs. Classification accuracy can be quite good even for small classification

Classifier, Features	Coarse $L(\mathbf{x})$	Fine $L(\mathbf{x})$	Overall $L(\mathbf{x})$	Overall $\mathcal{C}(\mathbf{x})$
Naive, $\phi_1(\mathbf{x})$	.139	.181	.229	**.168**
ACC, $\lambda_c = 1.5, \lambda_f = 1$	.140	.156	**.217**	.187
Naive, $\phi_1(\mathbf{x}), \phi_2(\mathbf{x})$	.128	.142	.200	.490
ACC, $\lambda_c = .1, \lambda_f = .075$	.111	.100	**.163**	**.431**
Naive, $\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \phi_3(\mathbf{x})$	.106	.108	.162	1.00
ACC, $\lambda_c = .02, \lambda_f = .02$	.108	.105	.162	**.691**

Table 4: Baseline results for misclassification and feature acquisition costs in a granular classification setting on Yahoo! Mail dataset, comparing progressive feature classes and cascade classifier, 10-fold cross validation. Bold values for overall within each group of rows indicate a significant difference at (p<.05).

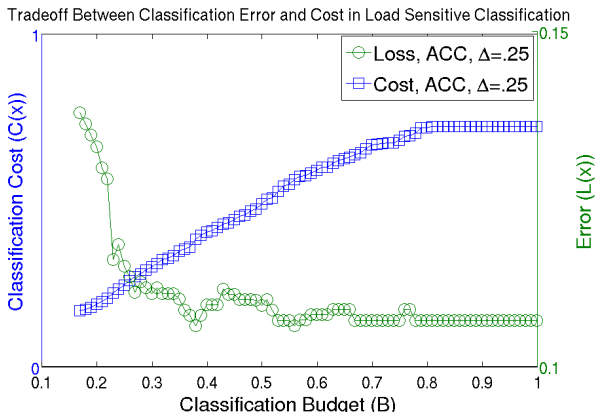


Figure 5: Tradeoff between Classification Error and Cost in Load-Sensitive Setting for the Yahoo! Mail dataset. Error can converge even with limited budgets

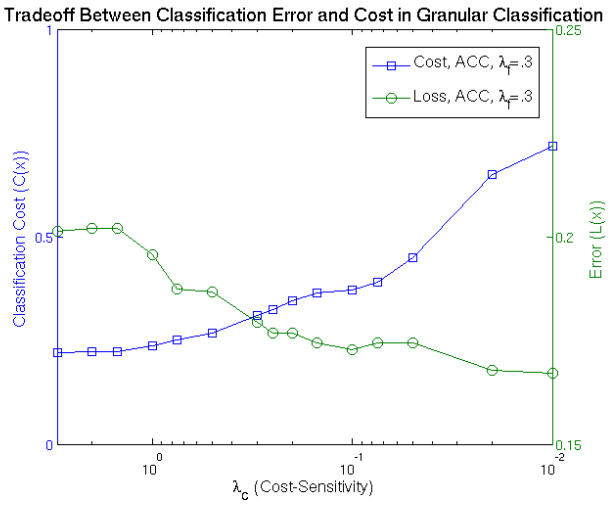


Figure 6: Tradeoff between Classification Error and Cost in Granular Setting for the Yahoo! Mail dataset. X-axis reversed

Classifier	$L(\mathbf{x})$	% $\mathcal{C}(\mathbf{x}) > B$	avg excess
Naive	.131	0.00%	0.00
ACC, $\Delta = 0$	.143	66.1%	.115
ACC, $\Delta = .25$	.110	17.1%	.002
ACC, $\Delta = .5$	.111	0.10%	.004

Table 5: Results for Load-Sensitive Classification task with different values of regularization constant, Δ for the Yahoo! Mail dataset, 10-fold cross validation

Classifier	$L(\mathbf{x})$	% $\mathcal{C}(\mathbf{x}) > B$	avg excess
Naive	.057	0.00%	0.00
ACC, $\Delta = 0$	.021	30.2%	.059
ACC, $\Delta = .25$	.024	0.00%	0.00
ACC, $\Delta = .5$	.027	0.00%	0.00

Table 6: Results for Load-Sensitive Classification ON TREC DATASET with different values of regularization constant, Δ for the TREC-2007 dataset, 10-fold cross validation

budgets, as shown in Figure 5.

Finally, the budget trade-off in Figure 5 for load-sensitive classification shows a parallel to the cost-sensitivity trade-off in granular classification shown in Figure 6. Using a budget, the cost of classifying a set of instances can be bounded, with lower accuracy at lower budgets. By adjusting the cost-sensitivity parameter to a cascade, the relationship between classifier cost and error is similarly regulated, with increased accuracy requiring increased costs. In granular classification, tuning the cost-sensitivity requires knowledge of the relative importance of error and cost. For load-sensitive classification, the budget for classification require similar consideration. Depending on the appropriate circumstance, Adaptive Classifier Cascades provide two approaches to achieving a necessary trade-off.

6. LIMITATIONS & FUTURE WORK

The common problem of minimizing computational costs at test time is well-studied, and many different solutions have been proposed[11][12]. In our work, we modify a popular solution to computation-constrained prediction, the classifier cascade, and apply this solution in two scenarios pertinent to e-mail classification. This modification of the classifier cascade requires base classifiers that provide a margin as a metric of the uncertainty of the base classifier's prediction. Our work does not explicitly model or exploit the margin distribution of the base classifiers, nor does it provide any

theoretical guarantees on the basis of these margins or the performance of the base classifiers. While having few restrictions provides a generalizable setting to explore solutions to the problem, one drawback is that the objective function we seek to minimize is discontinuous and non-monotonic. By including more restrictions on the margins and accuracy of the base classifiers, it may be possible to prove stronger results about the capabilities and performance of classifier cascades. We are working to form a better understanding of how the margins and accuracies of base classifiers can impact the behavior of cascades.

By virtue of the classifier cascade model and lack of restrictions on base classifiers, the optimization problem we consider is not suitable for gradient-based methods. Our use of grid search over the combinatorial parameter space is a simple global optimization, but there is extensive work on global optimization strategies for exact and approximate results. One area where better optimization models would be useful is the granular classification task, where each level of the classification hierarchy is currently optimized separately. A true optimum would require optimizing the parameters at every stage of the hierarchy in tandem, but for many problems the search space for these parameters is vast. We hope to explore algorithms that are able to approximate the global optimum in such problem settings in future work.

The first scenario we consider involves granular classification, or the prediction of a series of related labels in a label hierarchy. An obstacle to this approach is the lack of a universally accepted label taxonomy for e-mail messages. In fact, e-mail messages may possibly belong to multiple classes within a level of a taxonomy. A better approach might be to model a tag taxonomy and predict inclusion in a series of increasingly specific tags. While we have acquired a small dataset with granular labels, larger corpuses that include tag information are critical for continued research.

In the second scenario we consider, load-sensitive classification, the classifier cascade can adapt to quickly changing budgets. However, our solution requires explicitly regularizing using the variability in the instances during training time. Tuning this parameter may be difficult, and the true variability in test instances may be unknown or change in scenarios that generate high load. One solution to this problem would be calculating the weight of the regularization parameter online, and using a precomputed set of cascade parameters for the computed value of regularization. This option holds the possibility of adapting the method to an online, adversarial setting while maintaining the benefits of training offline, an area of research we are actively pursuing.

7. CONCLUSION

The use of classifier cascades for classifying e-mail messages shows great promise. Our work demonstrates that this approach can be used to balance trade-offs between cost and accuracy in different tasks. We introduce one scenario, **granular classification**, with the goal of predicting labels within a hierarchy where the cost-sensitivity varies with the level of the hierarchy. A second scenario we consider is **load-sensitive classification**, where classification is required within an arbitrary computational budget. We provide a novel formulation of classifier cascades using base classifiers trained on progressively increasing subsets of features. In our formulation, *Adaptive Classifier Cascades*, we parameterize the relationship between these base classifiers on the basis of classification margin, and then learn a policy consisting of parameters that minimize a combination of cost and error across training instances. Learning these parameters provides an intelligent feature acquisition strategy in classifying e-mail messages that can adapt to different tasks or different computational constraints. System administrators can use this approach to meet a variety of circumstances based on business needs of the organization. Using a real-world dataset, we provide a realistic model of feature costs. Our evaluation shows the applicability of classifier cascades to both scenarios. In granular classification, we can show significant reductions in cost, error, or both cost and error versus methods that use a fixed set of features for classification. We provide an analysis of error and cost in different operational conditions, showing that this approach is suitable for many possible trade-offs of cost and accuracy. For load-sensitive classification, we show that our method can classify instances for arbitrary cost budgets while reducing error relative to fixed-feature classifiers. By including a regularization parameter for modeling the variability in test instances, we improve results to provide stronger guarantees on the adherence to computational budgets. This initial work shows this method is especially well-suited to the problem of e-mail classification.

Acknowledgments

This work was partially supported by NSF Grant # IIS-0746930 and AFRL contract # FA8750-10-C-0191.

8. REFERENCES

[1] NIPS Workshop on Coarse-to-Fine Learning and Inference. `http://learning.cis.upenn.edu/coarse2fine/Main/HomePage`, 2010.

[2] X. Chai, L. Deng, Q. Yang, and C. X. Ling. Test-cost sensitive naive bayes classification. In *ICDM*, 2004.

[3] G. Cormack. Trec 2007 spam track overview. In *TREC*, 2007.

[4] H. Daumé III. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August 2004.

[5] J. Itskevitch. *Automatic Hierarchical E-Mail Classification Using Association Rules*. PhD thesis, Belorussian State Polytechnic Academy, 2001.

[6] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *ICML*, 1997.

[7] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In *ICDM*, 2004.

[8] J. Postel. RFC 821: Simple mail transfer protocol, Aug. 1982.

[9] V. Raykar, B. Krishnapuram, and S. Yu. Designing efficient cascaded classifiers: Tradeoff between accuracy and cost. In *KDD*, 2010.

[10] M. Saberian and N. Vasconcelos. Boosting Classifier Cascades. In *NIPS*, 2010.

[11] P. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *JAIR*, 2, 1995.

[12] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.