TEFE: A Time-Efficient Approach to Feature Extraction

Li-Ping Liu Yang Yu Yuan Jiang Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
{liulp, yuy, jiangy, zhouzh}@lamda.nju.edu.cn

Abstract

With the rapid evolution of internet applications, people all over the world are sharing pictures, videos and audios online, and thus, content-based analysis is often demanded. Test efficiency is crucial to the success of online information processing. One obstacle to high-speed testing is the time cost of feature extraction for test objects, particularly for objects with complex representation such as images, videos and audios. In this paper, we study the problem of reducing test time cost by extracting cheap but sufficient features. We propose the TEFE *(Time-Efficient Feature Extraction) approach, which balances between the test accuracy and test time cost by extracting a proper subset of features for each test object. In the implementation,* TEFE *trains a sequence of support vector machines and classifies each test object cascadingly. Empirical study shows that* TEFE *is time efficient while holding a classification accuracy close to that of using all features. It also shows that the test time is linearly adjustable in* TEFE.

1 Introduction

As a revolution of Internet usage, Web 2.0 applications encourage every Internet user to create, collaborate, communicate through the Web, such as Wikipedia, Facebook, YouTube, Picasa, and uncountable number of blog sites and forums. Web 2.0 has achieved great success in exchange and share of information within communities. Considering that the number of Internet users has exceeded 1.2 billion, a challenge here is that too much information is published every day, particularly in the form of images, videos and audios, to perform an efficient retrieval. Thus, in order to meet the information requirement of users, effective and efficient information retrieval techniques are urgently needed.

A key to the success of online information processing is a high speed of testing. In other words, results should be present to user as fast as possible. Previous research generally assume that the features of objects have already been extracted, and they try to achieve high-speed testing by selecting a learning algorithm to use, which is fast in making prediction, from a pool of learning algorithms that can be used to model the concerned objects. While the feature extraction cost is neglected. However, when an application involves complicate objects, such as images, audios and videos, it is often expensive in time cost to extract effective features from their raw appearances, which is one of the main obstacles to high-speed testing.

One direction towards a low test time cost is to design fast feature extraction approaches. Researchers have proposed many approaches to accelerate feature extraction process, such as fast image texture extraction algorithms [8, 12], direct extraction from image encodes [3], and parallel acceleration approach [4]. It is challenging to invent even faster features with much discriminative power.

Another direction of reducing test time cost is to reduce the features extracted from test objects. In many cases, a few features of an instance are sufficient to judge the label of that instance with high confidence. We call such instances as *easy* instances, and as the opposite, instances which require a lot of features for a correct classification are called as *hard* instances. Obviously it is not necessary to extract all features of easy instances, and thus the time of extracting expensive features of easy instances can be saved. On the surface this is a feature selection problem, but traditional feature selection approaches select the same subset of features for all instances. As the consequence, for easy instances there are still redundant features, while for hard instances some discriminative features are missed. A better way is to spend different efforts (time costs) on different instances, such that the overall test time cost is reduced in maximum subject to a condition that the classification accuracy is still acceptable.

In addition to reducing test time cost, a good approach should also be adjustable in the tradeoff between classification accuracy and feature extraction time cost. This is because that different users may have different preferences, e.g.., some users may want a higher accuracy while some other users may prefer a higher speed and tolerate a rela-

tively lower accuracy. A configurable approach would give user a better flexibility.

In summary, the desired approach should be able to

- *Extract test-efficient features:* For each test instance, it does not always extract all the features. Instead, it only extracts *sufficient but cheap* features, such that the features are sufficient for making a (almost) correct prediction, while the feature extraction time cost is as small as possible.
- *Make accurate classification:* Given a time budget for feature extraction, the test performance should be as high as possible.
- *Adapt to time budget:* The approach is able to adapt to different time budgets set by the user according to users' requirement, and the performance can improve if the time budget increases.

In this paper, we propose the TEFE (Time-Efficient Feature Extraction) approach for the above purposes. Unlike traditional approaches which extract all features, TEFE tries to extract only a portion of features for test objects. When a test object is received, TEFE first extracts features with the lowest costs. At this moment, if it is confident enough to predict the label of the object, a classification is made; otherwise, more features are extracted until the confidence reaches a pre-set threshold, or the time budget is over, or no more features are available. This strategy saves time of extracting expensive features for test objects which are easy to classify using simple features. The approach is flexible for user to express his/er preference on the tradeoff between the time cost and classification accuracy through setting the confidence threshold. Empirical study shows that TEFE is able to achieve a classification accuracy close to existing methods using a far less time cost.

The rest of this paper is organized as follows. Section 2 introduces some related work. Section 3 proposes the TEFE approach, followed by empirical evaluation in Section 4. Finally, Section 5 concludes.

2 Related Work

Traditional machine learning and data mining research assume that every misclassification causes the same cost. In real-world tasks, however, different misclassifications may cause different costs. So, cost-sensitive learning [6, 22] has attracted much attention. In addition to misclassification cost, various types of costs may be encountered [19], among which the *test cost* has been studied by many researchers. Roughly speaking, a test cost is the expense for doing a "test" to get a feature value. For example, in medical diagnosis, each feature is the result of a medical test (such as blood-test), and the price of the medical test is the test cost of the corresponding feature. The balance between the quality of the trained model and the cost of acquiring missing values in training set has been studied in [15]. The VOILA method [1] tries to select tests by exploiting the dependencies between missing features and information shared among different feature subsets. The PAS algorithm [17] considers both the acquisition cost and misclassification cost based on a decision tree. The csNB method [2] estimates the expected utility of a feature value in testing, and compares it with the cost for feature test. The SBT (Sequential Batch Test) algorithm [16] considers delayed test costs. Note that since test costs are mostly financial costs, these studies usually do not care time costs in estimating the utility of different tests, and thus we need new methods for the purpose of test-efficient feature extraction.

Feature selection [7] tries to remove irrelevant and/or redundant features to improve learning performance. Traditional feature selection techniques roughly fall into two categories, i.e., filter methods and wrapper methods. Filter methods use learner-irrelevant measurements to evaluate and select features, such as information gain and Relief [13]. Wrapper methods involve the final learner in the feature selection process, such as using the accuracy as the evaluation criterion for the goodness of features. Our work is different from traditional feature selection. First, we assume that different features are with different time costs, while traditional feature selection does not consider the differences in time costs. Second, we will use different features for different instances, while traditional feature selection will use the same subset of features for all instances. The most important difference is that, traditional feature selection assumes that there exist irrelevant and/or redundant features, while we do not. We assume that if considering the whole test set, all the features are useful; while if considering a specific test instance, maybe using a portion of features is sufficient. When a data set contains irrelevant and/or redundant features, we can either run TEFE directly or do feature selection at first and then run TEFE.

In [5], the JIT method is proposed and applied to spam detection. This method is a variant of ID3 decision tree. It estimates the utility of a feature using the linear combination of mutual information and time cost. Similar to other decision tree algorithms, JIT stops growing the tree when a node is pure or there is no more feature for a further partition. This method is suitable for categorical features. To deal with continuous features in our concerned applications, JIT has to resort to discretization. It is well-known that some useful information may be lost during discretization, which may influence the final accuracy. A special test-cost sensitive learning method, i.e., minimal cost decision tree [14], also suffers from the above weakness of JIT although it uses a different strategy to partition the tree nodes.

3 The TEFE Approach

Given a test object, ideally there exists a small subset of features which is sufficient for a correct classification. However, before any feature is extracted, we know nothing about the test object. An approach that meets our goal can take a greedy search strategy, such that it extracts one feature for the test object at each step, and then makes a decision on whether to extract one more feature or to make classification based on the extracted feature(s). Going to the next step to extract one more feature will take more time cost, but making classification will take the risk of misclassification caused by insufficient features extracted. The time cost of making classification should also be taken into account as a part of test time cost. Ideally, an algorithm should take the best choice in each step to minimize the overall expected test time cost, and to achieve an acceptable accuracy.

Figure 1 illustrates the greedy strategy. An oval indicates extracting a feature and making a choice between extracting one more feature and making the classification, where the subscripts indexes the extracted features. A diamond represents a base classifier, where the subscripts indicate features used by this classifier. Each time when a new feature is extracted, the strategy either makes classification based on all the extracted features if the confidence for classification is high enough, or extracts one more feature.

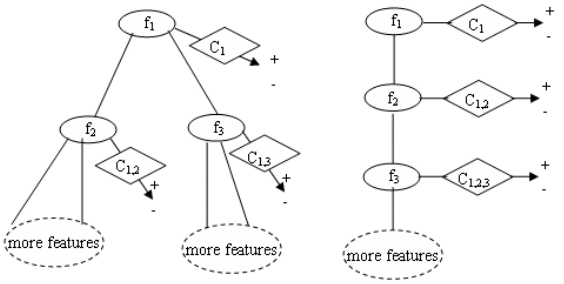

Figure 1. Illustration of the greedy strategy

There are two difficulties in implementing the above approach. First, there are many choices for the next feature to be extracted. Deciding which one to extract is not an easy task. Second, if the tree is fully grown, the number of base classifiers is exponential in the number of features. This will make the process infeasible.

One possible solution to overcome the difficulties is to prune the tree. There is an additional benefit of pruning, i.e., reducing the risk of overfitting. Similar to common decision trees, the deep nodes may contain only a few instances and it is not reliable to make further choices based on so limited information. Therefore, TEFE adopts a degenerated tree structure, as illustrated by the right part of Figure 1. Here, the feature extraction order is fixed, and the number of base classifiers equals to the number of features. Note that all the base classifiers can be generated from a training set, and they are ready to use before receiving the test objects.

In the training stage, TEFE ranks all features in an ascending list according to their feature extraction time costs, and then trains a sequence of base classifiers. The j-th classifier takes the first j features in the rank list. In other words, the j-th classifier is trained on the values of the first j features of the training set.

In the testing stage, TEFE extracts features along the rank list. Each time it obtains a feature value, it feeds all known feature values to the corresponding classifier. If the confidence reaches the threshold set by user, TEFE will take the prediction made by the classifier as the classification result; otherwise, TEFE will extract the next feature in the rank list, until no more features can be extracted. The threshold set by user is a parameter, which gives user the flexibility of balancing the classification accuracy and test time cost according to his/er favor.

3.1 The Training Stage

TEFE first ranks all features by their time costs. This ranking does not take the discriminability of features into account, since the discriminability of a feature is often hard to be measured, and when the time costs of the features span in a wide range, the factor of time cost will dominate the ranking. If there are many irrelevant and/or redundant features, feature selection can be executed at first. This will be discussed in Section 4.5.

Denote m as the total number of features, $\boldsymbol{X}$ as the training instances, $\boldsymbol{y}$ as their label vector, and n as the size of the training set. TEFE will train m base classifiers, each using one more feature than its predecessor. Here we use support vector machines with linear kernel for the base classifiers. The m SVMs, presented as $\{\text{SVM}_j = (\boldsymbol{w}_j, b_j)\}_{j=1}^{m}$, are to be obtained through Eq. 1, where $\boldsymbol{X}_i^{(j)}$ denotes the i-th instance with only the first j features.

$$
\begin{aligned}
(\boldsymbol{w}_j, b_j) = \arg\min \frac{1}{2}\|\boldsymbol{w}\|_2 + C\sum_i \xi_i \qquad (1)\\
s.t. \quad 1 - y_i(\boldsymbol{X}_i^{(j)}\boldsymbol{w} + b) \geq \xi_i\\
\xi_i \geq 0\\
i \in \{1, \ldots, n\}
\end{aligned}
$$

Note that SVM_{j+1} is trained on the same training examples as SVM_j except that one more feature is involved. Based on this recognition, we find that it is not necessary to train every SVM from scratch, and one SVM can serve as the starting point for its successor. Therefore, we have a

fast training process by solving the entire solution path of the SVM sequence.

The training process consists of m phases. In the first phase, SVM_1 is trained on the first feature. It is easy to get the solution analytically and we do not elaborate it here. In each following phase, say, the j-th, we solve the optimization problem in Eq. 2, where w_j is the j-th component of $\boldsymbol{w}$, corresponding to the newly added feature, and S is a variable constraining w_j. Eq. 2 is modified from the original QP problem of SVM with an additional inequality constraint.

$$\begin{aligned} \min \frac{1}{2}\|\boldsymbol{w}\|_2 + C\sum_i \xi_i \qquad (2)\\ s.t. \quad 1 - y_i(\boldsymbol{X}_i^{(j)}\boldsymbol{w} + b) \le \xi_i \\ \xi_i \ge 0 \\ |w_j| \le S \end{aligned}$$

At the beginning of the j-th phase, S is set to 0, and the optimal solution to Eq. 2 is $\boldsymbol{w} = (\boldsymbol{w}_{j-1}, 0)$ and $b = b_{j-1}$, where $(\boldsymbol{w}_{j-1}, b_{j-1})$ is the predecessor SVM. As we relaxing S, the optimal solution $\boldsymbol{w}$ and b will change and form a path. In the end, when S is relaxed to a value which does not constrain w_j, Eq. 2 becomes the optimization problem for SVM_j and the optimal solution $\boldsymbol{w}$ and b are exactly solution to SVM_j. Such a process connects one SVM with its successor. Repeating this process, all solutions of the SVM sequence can be obtained when the path grows from $j = 1$ to $j = m$. The detail is described as follows.

In the j-th phase, the path of $\boldsymbol{w}$ and b starts from the solution of SVM_{j-1} and ends at the solution of SVM_j. This path shows a piece-wise linear manner [10], that is, the path is composed of joint lines. On each line segment of the path, $\boldsymbol{w}$, b and the Lagrange multiplier $\boldsymbol{\alpha}$ are all linear in S. When a line ends, we only need to calculate the direction of the next line, until the phase ends. The direction of these joint lines and their ending points can be derived from Eq. 2.

Divide the training instances $\boldsymbol{X}$ into three sets, $\boldsymbol{X}_\mathcal{E}$, $\boldsymbol{X}_\mathcal{L}$ and $\boldsymbol{X}_\mathcal{R}$. The index sets $\mathcal{E}$, $\mathcal{L}$ and $\mathcal{R}$ are defined as

$$\begin{aligned} \mathcal{E} &= \{i : y_i(\boldsymbol{X}_i^{(j)}\boldsymbol{w} + b) = 1, 0 \le \alpha_i \le 1\} \\ \mathcal{L} &= \{i : y_i(\boldsymbol{X}_i^{(j)}\boldsymbol{w} + b) < 1, \alpha_i = 1\} \\ \mathcal{R} &= \{i : y_i(\boldsymbol{X}_i^{(j)}\boldsymbol{w} + b) > 1, \alpha_i = 0\}\ , \end{aligned}$$

where α_i is the multiplier of $\boldsymbol{X}_i^{(j)}$. $\mathcal{E}$ consists of indices of instances corresponding to support vectors. $\mathcal{L}$ consists of indices of instances inside the margin. $\mathcal{R}$ consists of indices of instances outside the margin.

The Lagrange primal problem of Eq. 2 is

$$\begin{aligned} C\sum_i \xi_i + \frac{1}{2}||\boldsymbol{w}||_2 + \sum_i \alpha(1 - y_i(\boldsymbol{w}\boldsymbol{X}_i^{(j)} + b)) \qquad (3)\\ - \sum_i \gamma_i\xi_i + \beta(|w_j| - S) \end{aligned}$$

Setting the derivatives to zero gives

$$w_{(j-1)} = \sum_i \alpha_i y_i \boldsymbol{X}_i^{(j-1)} \qquad (4)$$

$$w_j = \sum_i \alpha_i y_i \boldsymbol{X}_i^{j} - \beta\text{sign}(w_j) \qquad (5)$$

$$\sum_i y_i\alpha_i = 0 \qquad (6)$$

along with a KKT condition (only related ones are listed)

$$1 - y_\mathcal{E}(\boldsymbol{X}_\mathcal{E}^{(j)}\boldsymbol{w} + b) = 0\ . \qquad (7)$$

For convenience of calculation, differentiate Eqs. 4–7 with respect to β and get

$$dw_{(j-1)} = \sum_\mathcal{E} y_i \boldsymbol{X}_i^{(j-1)} d\alpha_i \qquad (8)$$

$$dw_j = \sum_\mathcal{E} y_i \boldsymbol{X}_i^{j} d\alpha_i - sign(w_j) \qquad (9)$$

$$\sum_\mathcal{E} y_i d\alpha_i = 0 \qquad (10)$$

$$y_\mathcal{E}(\boldsymbol{X}_\mathcal{E}^{(j)} d\boldsymbol{w} + db) = 0\ , \qquad (11)$$

where $w_{(j-1)}$ denotes the first $j-1$ dimensions of $\boldsymbol{w}$ while w_j denotes the j-th dimension of $\boldsymbol{w}$. By Eq. 5, w_j is calculated to be equal to $\sum_i \alpha_i y_i \boldsymbol{X}_i^j$, so $\text{sign}(w_j) = \text{sign}(\sum_i \alpha_i y_i \boldsymbol{X}_i^j)$. $\alpha_\mathcal{R}$ and $\alpha_\mathcal{L}$ do not change with S or β and $d\alpha_\mathcal{R}$ and $d\alpha_\mathcal{L}$ are zero, so they do not appear in this equation array. By solving Eqs. 8–11, we can get $d\boldsymbol{w}$, db and $d\alpha$, the direction of the next line.

At the beginning of each phase, $w_j = 0$, so $\beta = |\sum_i \alpha_i y_i \boldsymbol{X}_i^j|$ according to Eq. 5. As β decreases and the path grows, $\mathcal{E}$ will change when an instance i goes out of $\mathcal{E}$ (α_i reaches 0 or 1) or an instance i goes into $\mathcal{E}$ ($1 - y(\boldsymbol{X}_i^{(j)}\boldsymbol{w} + b)$ becomes 0). When this happens, the equation array does not hold any more and new directions need to be calculated with the updated $\mathcal{E}$. This is the very point where the current line ends and the next one starts.

When β decreases to 0, S does not constrain w_j and the Lagrange primal problem is the same as that of SVM_j, and $\boldsymbol{w}$ and b becomes the solution of SVM_j. Thus, the current phase ends.

When the m-th phase ends, all features have been involved and a sequence of m SVMs has been obtained. Figure 2 summarizes the whole training procedure.

Figure 3 illustrates the starting part of the solution path, where four features are involved. Each vertical line marks the end of a phase and the beginning of the next. The cross-points of $\boldsymbol{w}$, b and a vertical line is a solution of an SVM. A new component of $\boldsymbol{w}$ is added at every vertical line.

The training process is very efficient, because in each phase when a new feature is added, it can make good use

Figure 2. Pseudo-code of the training procedure

Input: Training set: $\boldsymbol{X}$ and $\boldsymbol{y}$
Output: A sequence of SVMs: SVM$_j$ $(j = 1, \cdots, m)$
Process:
  Generate SVM$_1$=$(\boldsymbol{w}_1, b_1)$ from $\boldsymbol{X}^{(1)}$;
  Initiate ;
  for $j = 2$ to m **do**
    $\boldsymbol{w} = (\boldsymbol{w}_{j-1}, 0)$, $b = b_{j-1}$, $\beta = |\sum_i \alpha_i y_i \boldsymbol{X}_i^j|$;
    while $\beta > 0$ **do**
      Solve the direction $d\boldsymbol{w}$, db, $d\ \ \boldsymbol{\varepsilon}$;
      Find the largest $\Delta\beta(\Delta\beta < 0)$ which causes $\mathcal{E}$ to change;
      $\boldsymbol{w} = \boldsymbol{w} + \Delta\beta d\boldsymbol{w}$, $b = b + \Delta\beta db$;
      $\varepsilon = \varepsilon + \Delta\beta d\ \varepsilon$, $\beta = \beta + \Delta\beta$;
      Update $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{L}$;
    end /*of while*/
    SVM$_j$ $= (\boldsymbol{w}, b)$
  end /*of for*/

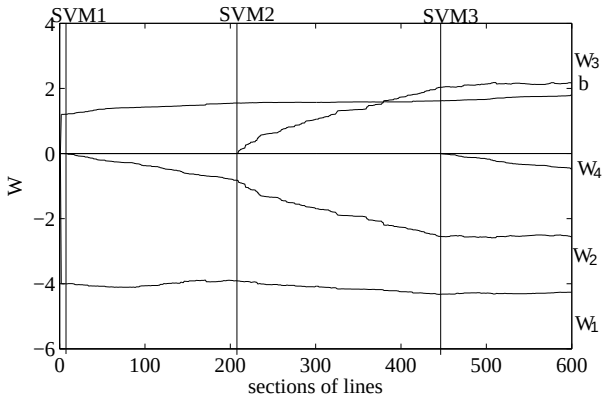

Figure 3. Illustration of the path of SVMs

of the result of the phase before. Another merit of this process is its high precision since it solves the problem analytically and the KKT condition is strictly satisfied at every step. Similar to [10], this method can be extended to use nonlinear kernels.

3.2 The Testing Stage

In the testing stage, TEFE tries to avoid the extraction of unnecessary features. The criterion of necessity is the classification confidence of the classifiers. Only when the confidence is lower than the threshold set by user, TEFE goes to extract one more feature.

The confidence of SVM on a test instance can be measured by the distance between the concerned instance and the decision boundary. The larger the distance, the more confident the classification. Here, a threshold thr is used to control the sanity check of the confidence. Upon receiving a new test instance, the first feature is extracted and then the first SVM is fired. If the distance is larger than thr, the current classification is regarded as the result; otherwise the next feature is extracted and the next SVM is invoked. Generally, for the i-th SVM, if Eq. 12 is satisfied, a classification is made without extracting any more features or invoking any more classifiers.

Figure 4. Pseudo-code of the testing procedure

Input: Test instance $\boldsymbol{x}$, threshold thr
Output: The label y for $\boldsymbol{x}$
Process:
  For $j = 1$ to m **do**
    Extract one feature for $\boldsymbol{x}$;
    Feed extracted features to SVM$_j$;
    if the classification margin of SVM$_j > thr$ or $j == m$
    then y = the classification given by SVM$_j$;
      break loop;
    end /*of if*/
  end /*of for*/

$$\frac{|\boldsymbol{x} \cdot \boldsymbol{w} + b_i|}{|\boldsymbol{w}|_2} \geq thr \tag{12}$$

Here the threshold thr provides the user a chance to control the tradeoff between test time cost and classification accuracy. The higher the threshold value, generally the higher the final classification accuracy and the more the features to be extracted. In many tasks the user may want to control the performance directly instead of through a parameter. However, the test accuracy is generally hard to foresee, and a direct control may be meaningless. For example, the user targets to 95% accuracy but the best achievable accuracy is only 90%. So, the soft-control on the performance accomplished by setting the parameter thr may be a better choice.

In real tasks, it is often the case that some features are extracted together. For example, image color histograms have several dimensions which are calculated together. We call such features as a feature group, and in the TEFE approach a feature group is regarded as a single feature.

Figure 4 summarizes the testing procedure. When a test instance comes, its first feature is extracted and fed to SVM$_1$. If the classification margin is larger than threshold thr, the test instance is labeled with the output of SVM$_1$; otherwise, the next feature is extracted and SVM$_2$ is fired. This process is repeated until no more feature is available, and in such case, the test instance is labeled by the last SVM.

For multi-class problems, one-vs-all decomposition can

Table 1. Image features in our empirical study

Feature description	# of Features	Time cost (sec. / image)
RGB of four central pixels of an image	12	4×10^{-4}
Mean hue of each 3×3 sub-images	9	5×10^{-3}
Histogram of hue in 16 bins	16	6.6×10^{-3}
Contrast feature [18]	1	8.3×10^{-2}
Directionality features [18]	4	4.0×10^{-2}
Coarseness features [18]	2	1.3×10^{-1}
Grayscale Gabor features [11]	12	$2.1 \sim 9.6 \times 10^{-2}$
Haralick texture features [9]	4	9.4×10^{-2}

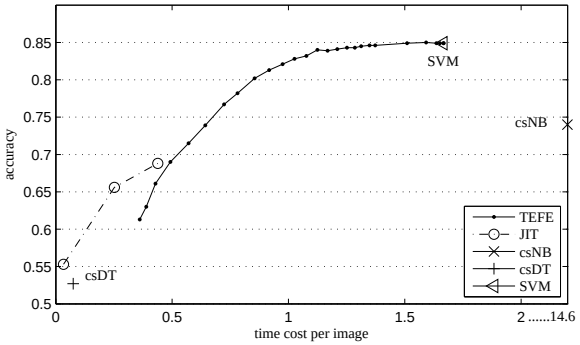

Figure 5. Average test time cost per image versus accuracy on scene classification

be used, and the difference only lies in that the confidence and prediction are given by multiple SVMs instead of a single SVM. For C classes we will train C sequences of SVMs, each sequence for one class. In the testing stage, when a new feature value is acquired, the extracted features will be fed to the corresponding SVMs in the C sequences. Here the stop criterion is that one of the C SVMs classifies the instance as positive with a confidence higher than the threshold thr, while the other SVMs classify the instance as negative; or no more feature is available. The remaining part is as same as that for two-class setting.

4 Empirical Study

4.1 Setting

Our empirical study involves several image applications. We employ several basic image feature extraction methods including color features and texture features, as tabulated in Table 1 where the running time per image (with size 384×256) of each method is also shown. All the time of executions are recorded on a machine with Pentium CPU 2.80GHz. These methods will generate sixty features in 28 groups for each image, where features in the same group are extracted together. Taking histogram of hue for example, when the histogram is calculated, all its 16 dimensions are obtained, or none can be calculated. The features of RGB values of sampling points, hue histogram, direction of Tamura features and Haralick features are grouped and must be calculated together. All the other features can be calculated one by one, among which one single feature is deemed as a group, thus there are 28 feature groups.

We compare TEFE with three test-cost sensitive algorithms, i.e., JIT [5], csNB [2] algorithm and csDT [14]. In addition, we also evaluate the performance of training SVM based on all features as a baseline. Since csNB has no parameters, its time cost and accuracy in classification is fixed. The JIT algorithm balances time cost and mutual information through its parameter α when splitting nodes on features. If α is set to 0, only mutual information is considered, and if α is set to 1, only time cost is considered. For JIT, we try different settings for its parameter α.

4.2 Comparison on Scene Classification

The dataset for scene classification is from [21], which contains 1,000 images of ten categories, and each category containing 100 images. Details of the data set can be found in [21]. Ten-fold cross-validation is executed on this dataset. For SVM and TEFE, the parameter C is set to 1.

By setting several different values for threshold thr, we get a curve of average test time cost per image versus accuracy for TEFE, as shown in Figure 5. By changing the parameter α, we also get a curve for JIT. The performance of each of the other three methods, i.e., csNB, csDT and SVM with all features (abbreviated as SVM), is shown as a marked point in Figure 5.

Figure 5 shows that TEFE has achieved a better performance than JIT, since 68.8% is the highest accuracy JIT can achieve when its parameter α is set to zero and the time cost is not taken into account. It is observed that JIT cannot utilize all features even when its parameter α is set to zero. In other words, the user cannot expect to utilize more features to achieve a better performance.

Compared with csNB, TEFE uses far less time to achieve the same accuracy. The accuracy of csNB is 74% with testing time 14.6 seconds per image. The test time cost is much more than that of TEFE, even much more than the time of extracting all features. This is because that the csNB method estimates the expected utility for every un-

Figure 6. Example images classified using different features. Left: Correctly classified using color features; Middle: Correctly classified using color and texture features; Right: Incorrectly classified even when all features are used

used feature in every round, which costs even more time than extracting the feature. Moreover, csNB does not provide the flexibility of controlling the tradeoff between test time cost and accuracy, and it will stop even when more features available.

Figure 5 also shows that TEFE is superior to csDT. The test time cost of csDT is very small, but it suffers from a very low accuracy (53%) that is not acceptable. This method does not provide flexibility of controlling the balance between test time cost and accuracy. Both csDT and JIT use tree-based classifiers, and therefore they suffer from similar deficiencies.

By comparing TEFE and SVM using all features, it can be found that when all features are extracted, their performances are the same. This is not strange because the classification made at the last round of TEFE is exactly made by a SVM using all features. In such case, the test time cost of TEFE increases by 0.002 second, which indicates that TEFE spends a little bit extra time. By decreasing the threshold of TEFE to a relatively small value, for example $thr = 4$, the test time cost of TEFE reduces to 0.763 second while the accuracy drops for only 4%, from 85% to 81%.

We also make an inner investigation of TEFE. We set the threshold thr to 3.6, which leads to an acceptable accuracy of 80.1%. In the first round, only the RGB values of four points are engaged in classification, and 36 of 100 images of the fourth class is labeled out and only 5 images from other classes are wrongly classified as this class. In this dataset, the fourth class *dinosaur* (an example is the top left image in Figure 6), is easy to separate from other classes, even by simply sampling four points from the center area of an image. This fact supports our motivation that simple features are sufficient for easy instances.

Table 2. Number of images classified in different round

Rounds	1st-11th	12th-27th	28th
Classified	347	349	304
Correctly classified	289	295	217

Table 2 shows the number of images labeled in different rounds. In the first eleven rounds, only color information is used, and then in the following rounds, texture features are involved. With color information only, 347 images are labeled, 83% of which are correct. This shows that color information alone can provide sufficient information for many instances, considering that the overall acceptable accuracy is 80.1%. The left column of Figure 6 shows two examples. Later, when texture features are involved, 349 more images are labeled. Two examples of such images are shown in the middle column of Figure 6. Finally, the most difficult 304 images go to the last round, among which 217 are correctly classified, with a relatively low accuracy. Two examples of such images are shown in the right column of Figure 6. Overall, we can conclude that TEFE has accomplished the task to extract simple features for easy instances and more features for difficult instances.

4.3 Comparison on Google Images

The second task is on data set obtained from Google image search. We selected 12 keywords and got 12 retrieval results. For each keyword, images of the first 200 results were downloaded. Note that the downloaded images are the resized images instead of the original images, so most of these images are less than 150×150 pixels. Images of each result are labeled as positive and negative, by considering whether they are related or not to its keyword. Images of this dataset are mostly smaller and more diverse than those in the scene classification dataset; this makes the classification task more difficult.

For images of every keyword, it is a binary classification problem and we use ten-fold cross-validation to evaluate the compared algorithms. The parameter C is set to 1 for SVM and TEFE. Results on the twelve sub-datasets are aggregated and shown in Figure 7.

Figure 7 shows that if all features are extracted, TEFE takes 0.18 second per image, but actually it can achieve a comparable performance using only 0.08 second. It is obvious that TEFE is superior to csDT and csNB. csNB costs 4.84 seconds per image, where a large portion of the time cost is spent to estimate the utility of features; but even with such a large time cost, its accuracy is only 62%. This might due to that its estimated utility is not very reliable. csDT is

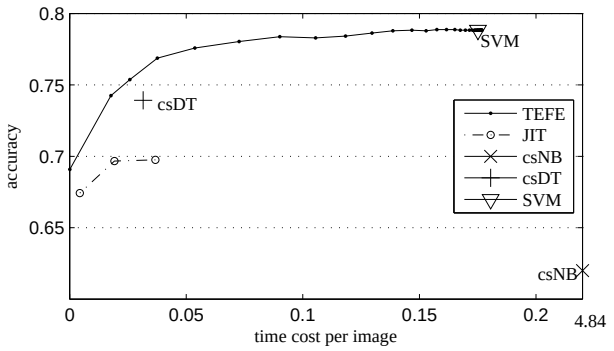

Figure 7. Average test time cost per image versus accuracy on Google images

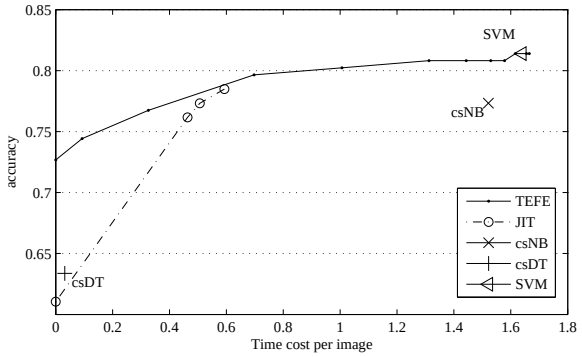

Figure 8. Average test time cost per image versus accuracy on synthetic images

better than csNB, but it is clear that with the same time, the accuracy of TEFE is better than that of csDT; and to achieve the same accuracy, TEFE uses a smaller time cost. Comparing TEFE and JIT it can be found that, by using the same test time cost, the accuracy of TEFE is much higher than that of JIT. Moreover, the best accuracy of JIT is 69.75% and it could not improve the performance further although there are more features that can be used, while TEFE is able to use more features to improve the performance further.

4.4 Comparison on Synthetic Images

Synthetic images, such as icons, maps, charts, etc., become more and more popular on the Internet. Fast and accurate classification of these images is increasingly important. The dataset used here is from [20]. The original dataset contains images from five classes and some of their labels are missing. We selected images from two classes, "map" and "artwork". The first class has 261 images while the second one has 256 images.

Results on this data set are shown in Figure 8. Similar to the results in previous subsections, the performance of TEFE is apparently superior to the compared methods.

4.5 Combination with Feature Selection

As mentioned before, feature selection can be applied when the data has irrelevant and/or redundant features. Here we use the scene classification dataset for experiments, with 67% instances for training and the remaining 33% for testing.

We first evaluate the traditional feature selection method. First, all 60 features are evaluated by Relief [13] and ranked by their merits on the training set. Then, every time the last feature in the rank is removed, a SVM is trained on the remaining features. The performance curve is shown as "Relief" in Figure 9. Then, we evaluate the performance of TEFE combined with feature selection. Here TEFE is applied to the data sets after the Relief feature selection procedure. We select 60 (i.e., no selection), 47, and 38 features according to the Relief rank, and then on the selected features we run TEFE. The performances are shown in Figure 9.

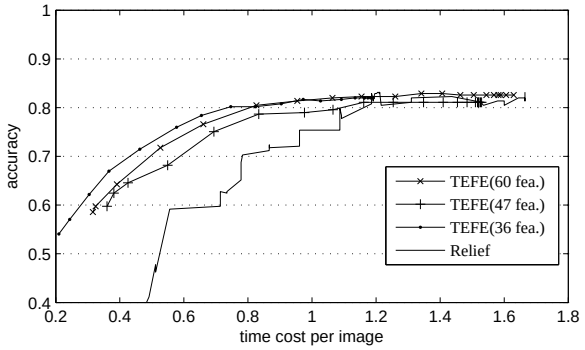

Figure 9. Average test time cost per image versus accuracy of TEFE with feature selection

From Figure 9 we can find that TEFE is beyond what have been done by traditional feature selection. The accuracy of TEFE without feature selection (i.e., TEFE with 60 features Figure 9) is always better than Relief. Moreover, in most part TEFE achieves the same accuracy of Relif with a much smaller time cost. This partially dues to the fact that Relif has thrown away some features which are helpful for classifying some specific images but the discriminability is weak if considering the whole dataset. Moreover, when the

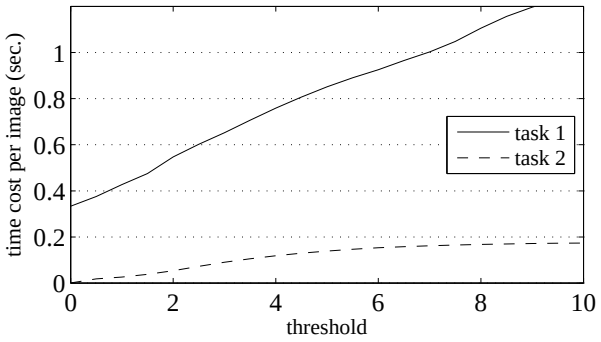

Figure 10. Threshold thr versus average test time cost

time budget is smaller, the difference between TEFE and Relif becomes more significant. For example, when the time budget is set to 0.8 second per image, the accuracy of TEFE reduces a little but the accuracy of Relife reduces about 10%.

It is noteworthy that by combining with a good feature selection process, TEFE is able to achieve a better performance than directly working on the original features (see TEFE with 36 features in Figure 9). However, the feature selection process has to be used carefully, because inappropriate feature selection will degenerate the performance (see TEFE with 47 features in Figure 9).

4.6 On the Adjustability

Different users may have different requirements, and even the same user may have different requirements under different situations. So, it is desirable that the balance between test time cost and classification accuracy can be controlled by the user. The adjustable parameter thr in TEFE provides such a flexibility to users.

Figure 10 plots the relationship between thr and test time cost on two data sets. It can be found that the relationship is almost linear. An initial explanation to this linear property is that, when thr increases by Δthr, a portion of instances p which were originally classified in each round, except the last round, are postponed to the next round. So, the time cost increased by $p \times \sum_{r=2}^{m} cost_r$, where r is index of the round and the slope is $p \times \sum_{r=2}^{m} cost_r / \Delta thr$.

This linear adjustability is a nice property of TEFE. For example, the linear relationship may be estimated by evaluating two sample thr values on the training set. Then, when the user allocates a test time budget, the system can estimate what is probably an appropriate thr for achieving the best accuracy within that time budget.

5 Conclusion

In many online applications, test time cost is a very important issue. Roughly, test time cost consists of two parts, i.e., time cost for feature extraction and time cost of making prediction. Most state-of-the-art high performance classifiers spend a small time cost in making prediction. Now, to reduce test time cost, more effort should be devoted to reducing feature extraction time cost.

In this paper, we propose the idea of extracting only "cheap but sufficient" features for test instances. For easy instance, instead of extracting all features, we try to extract only a few features which are with relatively small feature extraction time cost, but can provide sufficient for a correct classification.

To achieve this goal, we propose the TEFE approach. TEFE first orders the features according to their feature extraction time costs. For test instances, it extracts features one by one, and when the classification confidence based on the extracted features is high enough, the test instance will be classified and no more features will be extracted. A sequence of support vector machines are generated from training set, and there is an efficient solution to obtain those SVMs through regularization path. Experiment on three image applications show that the proposed TEFE method is superior to compared methods. It achieves acceptable high accuracy with a small test time cost, and provides the user with a flexibility to balance the test time cost and classification accuracy according to the user's favor. Moreover, it is possible to combine TEFE with feature selection methods to achieve even a better performance.

The current work can be extended in several directions. It is interesting to design a better method to rank features for our purpose. It is also interesting to incorporate ensemble methods into the framework to improve the exploitation of the extracted features.

Acknowledgements

This research was supported by the National NSFC (60505013, 60635030), JiangsuSF (BK2008018), FANEDD (200343), 863 Program (2007AA01Z169), and MSRA IST Program.

References

[1] M. Bilgic and L. Getoor. VOILA: Efficient feature-value acquisition for classification. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 1225–1230, Vancouver, Canada, 2007.

[2] X. Chai, L. Deng, Q. Yang, and C. X. Ling. Test-cost sensitive naive bayes classification. In *Proceedings of the 4th*

IEEE International Conference on Data Mining, pages 51–58, Brighton, UK, 2004.

[3] C. C. Chang, J. C. Chuang, and Y. S. Hu. Retrieving digital images from a JPEG compressed image database. *Image and Vision Computing*, 22(6):471–484, 2004.

[4] Y. Chung and V. K. Prasanna. Parallelizing image feature extraction on coarse-grain machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1389–1394, 1998.

[5] M. Dredze, R. Gevaryahu, and A. Elias-Bachrach. Learning fast classifiers for image spam. In *Proceedings of the 4th Conference on Email and Anti-Spam*, pages 487–493, Mountain View, CA, 2007.

[6] C. Elkan. The foundations of cost-senstive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 973–978, Seattle, WA, 2001.

[7] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[8] J. Haddadnia, M. Ahmadi, and K. Faez. An efficient feature extraction method with pseudo-Zernike moment in RBF neural network-based human face recognition system. *EURASIP Journal on Advances in Signal Processing*, 2003(1):890–901, 2003.

[9] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6):610–621, 1973.

[10] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[11] A. Jain and G. Healy. A multiscale representation including opponent color features for texture recognition. *IEEE Transactions on Image Processing*, 7(1):124–128, 1998.

[12] P. Janney, G. Sridhar, and V. Sridhar. Enhancing capabilities of texture extraction for color image retrieval. In *Proceedings of the 3rd World Enformatika Conference*, pages 282–285, Istanbul, Turkey, 2005.

[13] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 122–134, San Jose, CA, 1992.

[14] C. X. Ling, Q. Yang, J. Wang, and S. Zhang. Decision trees with minimal costs. In *Proceedings of the 21st International Conference on Machine Learning*, pages 69–76, Banff, Canada, 2004.

[15] P. Melville, F. Provost, M. S. Tsechansky, and R. Mooney. Economical active feature-value acquisition through expected utility estimation. In *Proceedings of the 1st Workshop on Utility-Based Data Mining*, pages 10–16, Chicago, IL, 2005.

[16] V. S. Sheng and C. X. Ling. Feature value acquisition in testing: a sequential batch test algorithm. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 809–816, Pittsburgh, PA, 2006.

[17] V. S. Sheng and C. X. Ling. Partial example acquisition in cost-sensitive learning. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 638–646, San Jose, CA, 2007.

[18] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8(6):460–473, 1978.

[19] P. D. Turney. Types of cost in inductive concept learning. In *Proceedings of the ICML'2000 Workshop on Cost-Sensitive Learning*, pages 15–21, Stanford, CA, 2000.

[20] F. Wang and M. Y. Kan. NPIC: Hierarchical synthetic image classification using image search and generic features. In *Proceedings of the 5th International Conference on Image and Video Retrieval*, pages 473–482, Tempe, AZ, 2006.

[21] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.

[22] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. In *Proceeding of the 21st National Conference on Artificial Intelligence*, pages 567–572, Boston, WA, 2006.