Improving Semantic Concept Detection through Optimizing Ranking Function

Sheng Gao* and Qibin Sun

Abstract—**In this paper, a kernel-based learning algorithm, *KernelRank*, is presented for improving the performance of semantic concept detection. By designing a classifier optimizing the receiver operating characteristic (ROC) curve using *KernelRank*, we provide a generic framework to optimize any differentiable ranking function using effective smoothing functions. *KernelRank* directly maximizes a one-dimensional quality measure of ROC, i.e. AUC (Area under the ROC). It exploits the kernel density estimation to model the ranking score distributions and approximate the correct ranking count. The ranking metric is then derived and the learnable parameters are naturally embedded. To address the issues of computation and memory in learning, an efficient implementation is developed based on the gradient descent algorithm. We apply *KernelRank* with two types of kernel density functions to train the linear discriminant function and the Gaussian mixture model classifiers. From our experiments carried out on the development set for TREC Video Retrieval 2005, we conclude that (1) *KernelRank* is capable of training any differentiable classifier with various kernels; and (2) the learned ranking function performs better than traditional maximization likelihood or classification error minimization based algorithms in terms of AUC and average precision (AP).**

Index Terms—**Information Retrieval, Multimedia Database, Semantic Concept Detection, ROC curve, Area under ROC**

I. INTRODUCTION

In a ranking system, the ranking function is utilized to sort the samples in a database according to their relevant degrees to the query so that only highly relevant documents are shown to the user. For example, in the high-level feature extraction in TREC Video Retrieval (*TRECVID*), the system returns the top-N video shots for a given semantic concept. In general, the ranking efficiency is measured using the ranking metrics, e.g. AUC (Area under the ROC curve) or AP (non-interpolated average precision). The objective in the paper is to investigate an efficient learning algorithm for designing the ranking function to optimize the ranking performance.

Similar to supervised learning, a ranking function is estimated from a given set of training samples labeled as relevance (*positive*) or irrelevance (*negative*). Usually, learning of the ranking function can be formulated as a binary classification problem, so that the documents are sorted according to the output of the classifier. Thus, any classifier, such as support vector machine (SVM), linear discriminant function (LDF), Gaussian mixture models (GMM) etc., is applicable [5]. However, traditional learning algorithms train the classifiers by minimizing the classification error or maximizing the likelihood rather than maximizing the ranking metrics which is utilized to evaluate the system of semantic concept detection. Such criterion mismatch definitely affects the ranking performance. It is even worse when the dataset is highly imbalanced. Cortes & Mohri [3] studied the relationship among the ratio of class distribution, classification error rate and the average AUC. They experimentally showed that the average AUC coincides with the accuracy only in the case of even class distribution, where the AUC monotonically increases with accuracy. Their analysis provided a reasonable explanation for the partial success of applying classifiers learned for minimizing classification error to ranking. For example, SVM trained for minimizing classification error is widely exploited for multimedia semantic concept detection in TRECVID. Nevertheless, they also pointed out the high variance of AUC is observed when the class distributions are uneven. This implies the classifier having a fixed classification error rate demonstrates a noticeable difference of AUC for the highly uneven class distribution. The uneven distribution is a critical issue occurred in information retrieval and data mining. In most real classification (or ranking) problems, the negative samples are much more than the positive samples and the ratio between them varies much across different datasets. For example, it is often seen that there are only a few hundred positive samples versus thousands of negative samples. Therefore, it is desirable to find a new learning criterion, rather than classification error rate, to design an optimal ranking system.

AUC characterizes the correct rank statistics on a dataset by a given utility function. Thus optimizing AUC is a candidate criterion to design the ranking function. The AUC-based objective function is a function of the scores calculated from the classifier estimated on training samples. Therefore, the parameters of the classifier are naturally embedded into the AUC definition. Maximizing the AUC metric will result in a classifier that provides the maximal ranking performance.

Like all other metrics such as classification error rate, recall, precision or F_1 measure, AUC is also a discrete metric which measures how much proportion of pair-wise samples between the positive and the negative is correctly ranked. To solve the AUC-based objective function, smoothing is first applied in

Manuscript received July 6, 2006; revised June 3, 2007.

Sheng Gao is with the Institute for Infocomm Research, A*STAR, 21 Heng Mui Keng Terrace, Singapore 119613 (corresponding author: 65-68748531; fax: 65-67744998; e-mail: gaosheng@i2r.a-star.edu.sg).

Qibin Sun is with the Institute for Infocomm Research, A*STAR, 21 Heng Mui Keng Terrace, Singapore 119613 (e-mail: qibin@i2r.a-star.edu.sg).

order to obtain a differentiable function. In the paper, an efficient kernel-based learning algorithm, *KernelRank*, is presented to design the classifiers that have the optimal ranking performance. First, *KernelRank* models the distribution of ranking scores using the kernel density estimation (i.e. *Parzen window*). Then, the AUC is calculated using the integral of the score distribution. Thus, a differentiable objective function is derived. Finally, the parameters of the classifiers are estimated using the gradient descent algorithm. Kernel density estimation (*KDE*) allows us to approximate ranking loss using the various functions instead of the sigmoid function [11, 12, 22]. *KernelRank* provides a generic and flexible framework for learning the ranking function. To address the issues of the high cost of computation and memory due to the pair-wise interactions, an efficient implementation, which is based on the gradient descent algorithm, is introduced. Two specific ranking functions, i.e. LDF and GMM, are learned using the *KernelRank* algorithm and evaluated on the large-scale development set used in TRECVID 2005.

The paper is organized as follows. Related work is studied in the next section. Then the details of designing the objective function using kernel density estimation are given in Section III. In Section IV, the implementation and estimation of *KerneRank* algorithm is described. The experimental evaluation and analyses are presented in Section V. Finally, we summarize our findings in Section VI.

II. RELATED WORK

Using the ROC curve to evaluate the system performance has been extensively studied in the community of machine learning [4, 7, 9]. Unlike the widely used metrics such as classification error rate, precision or F_1 measure, which only measure the behavior of the classifier at one chosen decision point, the ROC is an overall measure. Designing a classifier with an optimal ROC is preferred, particularly when we have little knowledge about the problems in hand. For instance, sometimes the cost of false decision is unknown or changing and the ratio of class distributions is not predictable. Traditionally, the classifier is trained on the assumption of prior knowledge, e.g. the costs of false decision for all classes are equal and the class distribution is even and constant over time. When the real situation does not follow these assumptions, the system performance is degraded. In the semantic concept detection problem, the assumptions of equal cost and fixed class distribution are not valid. The ratio between the negative samples and the positive samples is large and varies across the datasets and concepts, as will be shown in Section V. It implies that the cost of misclassifying a positive sample should be higher than that of misclassifying a negative sample and the cost should be different over the concepts and datasets. Therefore, it is necessary to develop a learning algorithm to handle such an issue. Since the ROC is a measure that is independent of the cost and the ratio of class distributions, optimizing the ROC would be a good exploration.

Recently, many works have been done to study the learning problem. These works learn the classifiers through maximizing the AUC, a one-dimensional quality measure of the ROC curve. One way to realize the optimization is to minimize the pair-wise classification error, a value equivalent to one minus the AUC. In [3], a theoretical analysis is presented to study the relationship between AUC optimization and classification error minimization. In [10], RankBoost is developed to learn a set of weaker classifiers which minimize the pair-wise ranking error. In [22], a margin-based bound for ranking is presented and a smooth margin ranking algorithm is proposed. Applying SVM to AUC maximization is studied in [21], where the objective is to minimize the penalized pair-wise ranking error under a set of pair-wise constraints. Joachims studied the application of SVM for multivariate performance measures (e.g. precision, F_1, AUC, etc.) in [15], where learning SVM for maximizing the AUC is a special case. Another way is to directly maximize the AUC-based objective function that is smoothed using some approximation functions. Then the highly non-linear function is solved using the gradient descent algorithm [2, 13, 23]. These works usually use the sigmoid function for smoothing and they don't study the effects of smoothing functions on the ranking performance. Other methods include updating the decision tree for AUC maximization. For example, Ferri et al. [8] used the AUC as a splitting criterion to build the tree, while Ling & Yan presented a probability estimation algorithm [18]. In these works, the objective function is the AUC measure. Therefore, this kind of learning is related to the MFoM algorithm, where the objective function is an approximated metric such as F_1 measure used for evaluation [11, 12]. In our previous work, we have discussed an ensemble approach for ROC optimization [25], where a collection of classifiers are designed and each of them is optimized at the selected point in the ROC curve. However, the smoothing function is still sigmoid.

The presented work, *KernelRank*, follows the latter path, i.e. training the classifier through maximizing the smoothed AUC metric based on kernel density estimation. It is a further extension of our work on this issue [24], where *KernelRank* is outlined without the in-depth discussion and experimental analysis. In the paper, we will give a thorough study on how to utilize the KDE to design a suitable ranking algorithm and what effect of KDE is on the ranking performance. Designing smoothing functions to approximate ranking error have not been studied up to now. Traditionally only a few well-known functions such as the sigmoid are applied. Thus, the ranking performance should depend on it. Recently, we noted Rudin [26] applied a ℓ_p-norm function for smoothing so that the top documents in the ranked list are emphasized.

III. KERNEL-BASED AUC OPTIMISATION

Learning a classifier for ranking is defined as follows. Given a set of training samples, T, having M positive samples and N negative ones, learn a binary classifier, $f(X|\Lambda)$, with the parameter set Λ so that the function gives a higher score for the positive samples than for the negative ones. We denote X as a feature representation of the sample and X^+ for the positive sample and X^- for the negative one. S^+ and S^- are the values of $f(X^+|\Lambda)$ and $f(X^-|\Lambda)$, respectively.

A. AUC Metric

The AUC, as the quantity of ranking performance for a classifier, is defined as the probability of the positive samples ranked higher than the negative ones, i.e.,

$$U = P(S^+ > S^-) \tag{1}$$

If the joint probability density distribution $g(S^+, S^-)$ of the positive score S^+ and the negative one S^- is available, then the exact value of Eq. (1) can be calculated using the integral of $g(S^+, S^-)$ on S^+ and S^-. However, in real applications, not only the probability density function of the scores is unknown, but also the scores are calculated from the classifier with unknown parameters. The empirical estimation of AUC for a known classifier is calculated on the training samples as:

$$U = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} I(S_i^+, S_j^-) \tag{2}$$

where $I(S_i^+, S_j^-)$ is an indicator function. It is equal to one when the pair-wise ranking is correct, i.e. $S_i^+ > S_j^-$, and zero otherwise.

B. Approximating AUC with Kernel Density Estimation

Eq. (1) and (2) are embedding functions of the classifier parameters. Maximizing it generates a classifier with the optimal ranking measure. However, the function is not differentiable. It is necessary to smooth it before optimization. Hereinafter, we will talk about designing a differentiable function to approximate the AUC in Eq. (2).

First, we discuss the estimation of the joint distribution $g(S^+, S^-)$. It can be estimated using the parametric statistical model or non-parametric kernel density estimation. Herein we adopt the latter and estimate it from the training samples. Assuming that S^+ and S^- are independent and have the density distributions $g^+(S)$ and $g^-(S)$, respectively, then $g(S^+, S^-)$ is factored into $g^+(S).g^-(S)$. When the training set and the classifier are ready, a set of score samples for the positive samples, S^+, and the negative samples, S^-, are collected. Therefore, the individual density distributions, $g^+(S)$ and $g^-(S)$, are empirically estimated using the kernel density function $K(S_1,S_2)$ as follows:

$$g^+(S) = \frac{1}{M}\sum_{i=1}^{M} K(S_i^+, S) \tag{3}$$

$$g^-(S) = \frac{1}{N}\sum_{i=1}^{N} K(S_i^-, S) \tag{4}$$

where $\int K(S_i^c, S)\, dS = 1, c = \{+,-\}$.

Thus, the cumulative density function (*CDF*) $G(Z)$ of the variable $Z = S^- - S^+$ is an integral of $g(S^+, S^-)$ on the variables S^+ and S^-. Similarly, in Rudin et al [22], Z is a measure of the ranking margin. Given a positive-negative sample pair, Z is less than zero for the correct ranking and larger than zero for the wrong ranking. For the correct ranking, we expect the classifier to have Z values much larger than zero. The *CDF* is calculated as:

$$G(z) = P(Z < z) = \iint_{S^- - S^+ < z} g^+(S^+).g^-(S^-)\, dS^+ dS^- \tag{5}$$

Here z is a constant acting as the threshold for ranking decision. Substituting Eqs. (3-4) into Eq. (5), we can get the empirical estimation of $G(Z)$ as:

$$G(z) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} \Phi(S_i^+, S_j^- | z) \tag{6}$$

$$\Phi(S_i^+, S_j^- | z) = \int_{-\infty}^{+\infty}\int_{-\infty}^{S^+ + z} K(S_i^+, S^+)\cdot K(S_j^-, S^-)\, dS^- dS^+ \tag{7}$$

For simplicity, we assume the variable values, S^+ and S^-, are within the range between negative infinity and the positive infinity. When their values are in other range, it is easy to derive the similar forms for Eqs. (6-7).

Eq. (7) is a smoothed version of the indicator function, $I(S_i^+, S_j^-)$, which depends on the value z. Obviously, the AUC defined in Eq. (2) is a special case of Eq. (6) when $z=0$ and the heavisible step function is applied in Eq. (7). Thus, we have,

$$U = G(0) \tag{8}$$

When a suitable kernel function is chosen, the smoothed indicator function in Eq. (7) can be efficiently computed. Thus, the objective function for AUC maximization can be defined for any interested classifier.

Next, we discuss two possible instances of the smoothed indicator function designed using the kernel density estimation.

1) Gaussian kernel density function

The first case is to use the Gaussian kernel density function for approximation. The Gaussian kernel is defined as:

$$K(x, y) = \frac{1}{(2\pi)^{1/2}\sigma}\exp\left(-(x-y)^2 / 2\sigma^2\right) \tag{9}$$

where σ is a variance to control the size of smoothing window. Substituting it into Eq. (7) and we get a Gaussian kernel-based approximation function,

$$\begin{aligned}\Phi(S_i^+, S_j^- | z) &= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty}\exp\left(-x^2/2\sigma^2\right) \\ &\quad \cdot\int_{-\infty}^{x+z_{ij}+z}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-y^2/2\sigma^2\right)dydx \\ &= \int_{-\infty}^{z_{ij}+z}\frac{1}{\sqrt{2\pi}\sqrt{2}\sigma}\exp\left(-y^2/4\sigma^2\right)dy\end{aligned} \tag{10}$$

with $z_{ij} = S_i^+ - S_j^-$. There is no analytic solution for it. However, it does not affect the learning algorithm used in the paper because its gradient over z_{ij} is analytic (see Section IV). The differentiable function has two variables, i.e. the pair-wise score difference between the positive and the negative samples, z_{ij}, and the margin value, z. The score is the output of the classifier, thus, the model parameters are naturally embedded into Eq. (10) and the AUC-based objective function is defined as:

$$U = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\Phi\left(S_i^+, S_j^- \middle| z=0\right) \tag{11}$$

with $S_k^c = f\left(X_k^c \middle| \Lambda\right)$, $c=\{+,-\}$, for the *k*-th training sample. In Eq. (11), only the parameter set Λ is unknown and needs to be estimated on the training set. Many optimization algorithms can be utilized to solve the equation.

2) Sigmoid-like kernel density function

The second kernel, a sigmoid-like density function whose cumulative function is a sigmoid one, is defined as:

$$K(x,y) = \alpha l_{xy}\left(1-l_{xy}\right) \tag{12}$$

with

$$l_{xy} = 1\Big/\left(1+\exp\left(-\alpha\left(x-y\right)\right)\right) \tag{13}$$

It is easy to derive the smoothing function, $\Phi\left(S_i^+, S_j^- \middle| z\right)$, for the kernel:

$$\Phi\left(S_i^+, S_j^- \middle| z\right) = \frac{x}{(x-1)} - \frac{x}{(x-1)^2}\cdot \ln x \tag{14}$$

with $x = e^{\alpha\left(z_{ij}+z\right)}$. At the discontinuous point $x=1$ (i.e. $z_{ij}+z=0$), its value is equal to its limit, *0.5*. Similar to the case of Gaussian kernel, the AUC-based objective function Eq. (11) can be obtained by substituting Eq. (14) into it.

C. Experiential AUC curves

It is expressive and useful to visualize the AUC curve. The AUC is a random variable that is further interlinked with the parameters of the classifier (see Eq. (1)). It is impossible, in practice, to know its probability density function (*PDF*) or CDF. Therefore, the experiential AUC curve is derived for different kernels based on the samples (see Eq. (2) and its variable smoothing versions such as Eq. (11)).

The approximated AUC definition in Eq. (11) is the function of the pair-wise score difference variable, z_{ij}, between the positive and negative samples and summed over all possible pairs. If we make the following assumptions: (1) z_{ij}'s are random variables and independent of each other and (2) they have the same probability density distributions *pdf(z)*, then each term in the sum has the same distribution which makes the AUC random variable have a simple form of *z*. Therefore, the *PDF* of AUC can be derived.

To plot the experiential AUC curves, *pdf(z)* is estimated using the samples of pair-wise score difference from our experiments. Figure 1 shows the experiential CDF curves of the AUC for Gaussian kernel (marked *GAUS*, diamond dark line), sigmoid-like kernel (marked *SIGL*, empty circle pink line), sigmoid function (marked SIG, triangle red line), and the non-smoothing curve (marked *Discrete*, cross blue line). The samples for drawing the curves are randomly selected from one experimental result on the evaluation set for the concept *Building* (TRECVID 2005) with the textual modality feature (see Section V for details). There are 143 positive score examples and 857 negative ones selected so that 122,551 different score pairs are generated. The variance coefficient for the Gaussian kernel and alpha values for the sigmoid-like kernel and sigmoid smoothing function are set according to our experiments. The X-axis is the pair-wise score difference and the Y-axis is the AUC value. It is found that the AUC curves for the three smoothing functions are very similar.

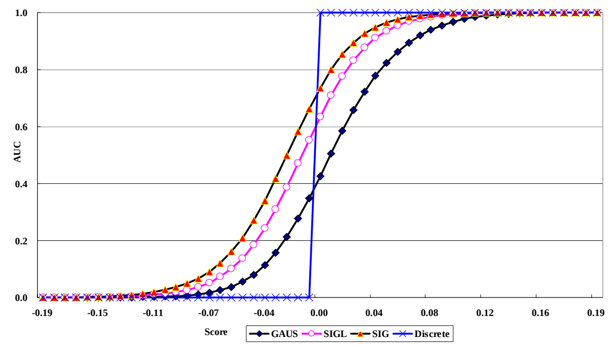
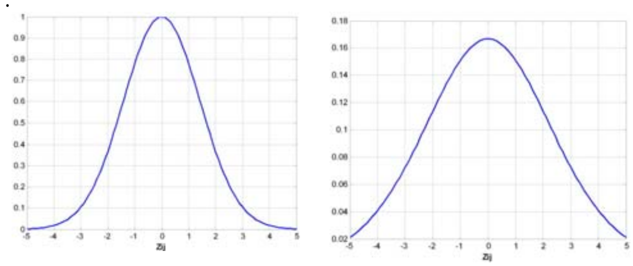


Fig.1. Experiential AUC curves for Gaussian kernel (diamond dark line), sigmoid-like kernel (empty circle pink line), sigmoid smoothing (triangle red line), and non-smoothing (cross blue line) (X-axis: the value of Z_{ij}. Y-axis: AUC value. Variance in Gaussian kernel: 0.043. Alpha in Sigmoid-like kernel and Sigmoid: 7.45)

.



(a) (b)

Fig.2. First-order derivative of the smoothed indicator functions for AUC approximation (z=0), (a) Gaussian kernel function ($\sigma=1.0$); (b) Sigmoid-like kernel function ($\alpha=1.0$). X-axis: the value of Z_{ij}. Y-axis: the value of the gradients.

IV. LEARNING WITH GRADIENT DESCENT ALGORITHM

As the indicator function is smoothed using the kernel methods discussed above, the AUC-based objective function is now differentiable for optimization. By maximizing the objective function, the parameters of the classifier are estimated. The objective function is often highly non-linear for the chosen classifier. Its solution is calculated using the gradient descent algorithm, thus the solution is locally rather than globally optimal. Furthermore, the starting point in the iterations affects the estimation. In our experiments, we start the iterative algorithm from the point estimated using the traditional maximum likelihood (*ML*) algorithm.

A. Parameter Estimation

We exemplify the *KernelRank* estimation algorithm using the kernel functions introduced in Section III. The objective function is defined in Eq. (11) and its gradients with respect to Λ can be derived as:

$$\nabla U\big|_{\Lambda} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\nabla\Phi\left(S_i^+, S_j^- \middle| z=0\right)\Big|_{z_{ij}}\cdot\left(\nabla f\left(X_i^+\middle|\Lambda\right)\Big|_{\Lambda} - \nabla f\left(X_j^-\middle|\Lambda\right)\Big|_{\Lambda}\right) \tag{15}$$

The first term in the summary is the gradient with respect to

z_{ij}, a variable of pair-wise score difference between the positive sample and the negative one. Its form depends on the kernel. The second term, whose form depends on the classifier, is the difference of gradients between the positive sample and the negative one. Thus the kernel dependent part and the classifier dependent part are separated.

For the Gaussian kernel, the first term is:

$$\nabla\Phi\left(S_i^+,S_j^-\middle|z=0\right)\Big|_{z_{ij}}=\frac{1}{\sqrt{2\pi}\sqrt{2}\sigma}\exp\left(-z_{ij}^2\Big/\left(2\left(\sqrt{2}\sigma\right)^2\right)\right) \tag{16}$$

while for the sigmoid-like kernel, it is:

$$\nabla\Phi\left(S_i^+,S_j^-\middle|z=0\right)\Big|_{z_{ij}}=\alpha x\left(\frac{x+1}{(x-1)^3}\cdot\ln x-\frac{2}{(x-1)^2}\right) \tag{17}$$

with $x=e^{\alpha\cdot z_{ij}}$. At the discontinuous point $z_{ij}=0$, the value of Eq. (17) equals to *1/6*. Figure 2 shows the curves for the above two gradient functions.

The second term will be determined once the classifier is designated. Using the above gradients, the iterative algorithm is utilized to seek the parameters of classifiers.

B. Efficient Implementation

To get the gradient of a parameter from Eq. (15), it requires *M*N* sums and *M*N* pair-wise gradients over high dimensional model parameters. For example, in our experiments with the 3,464-dimensional textual feature for the concept *Building*, the number of pairs is about $3.6*10^7$ (i.e. 2,008*17,935) and the dimension of the model parameters is 3,464. It is a huge burden for computation and memory. Some efforts have been done to reduce the burden by selecting a few numbers of neighboring samples for a given sample rather than all pairs for approximate computation [13, 21]. However, the approximation computation is not necessary in our algorithm. Exact computation can be efficiently performed.

We reorganize Eq. (15) as:

$$\nabla U\big|_\Lambda=\frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{N}\sum_{j=1}^{N}\nabla\Phi\left(S_i^+,S_j^-\middle|z=0\right)\Big|_{z_{ij}}\right)\cdot\nabla f\left(X_i^+\middle|\Lambda\right)\Big|_\Lambda$$
$$-\frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{M}\sum_{i=1}^{M}\nabla\Phi\left(S_i^+,S_j^-\middle|z=0\right)\Big|_{z_{ij}}\right)\cdot\nabla f\left(X_j^-\middle|\Lambda\right)\Big|_\Lambda. \tag{18}$$

The inner sum in the first line of Eq. (18) is the average gradient on z_{ij} of the smoothed indicator function for a positive sample summed over all negative samples. Its value indicates the contribution degree of the positive sample to the overall gradient, which weighs the gradient of the classifier $\nabla f\left(X_i^+|\Lambda\right)$. The outer sum is a weighted average of gradients $\nabla f\left(X_i^+|\Lambda\right)$, computed on all positive samples. Similarly, the inner sum in the second line measures the contribution degree for a negative sample while the outer sum is the weighted average of gradients summed on the negative samples.

The reorganization reduces the number of gradients to be calculated for the classifier from *M*N* to *M+N*. However, there are still *M*N* gradients to be calculated for the smoothing function. In practice, the latter has much lower computation cost when compared with the former considering that the size of classifier parameters is high (e.g. 3,464 for textual feature) while the latter only needs to calculate the gradient for a one-dimensional variable z_{ij}.

TABLE 1
DETAILED DESCRIPTION OF THE EXPERIMENTAL DATASET DERIVED FROM THE DEVELOPMENT SET OF TRECVID 2005

CONCEPT	TEXTUAL	VISUAL
BUILDING (C1)	T: 19,943 (2,008) V: 8,538 (1,254) E: 6,447 (958)	T: 41,978 (3,604) V: 11,173 (1,416) E: 8,295 (1,064)
CAR (C2)	T: 19,943 (1,204) V: 8,538 (624) E: 6,447 (272)	T: 41,919 (2,253) V: 11,325 (767) E: 8,487 (370)
EXPLOSION_FIRE (C3)	T: 19,943 (492) V: 8,538(71) E: 6,447 (23)	T: 42,038 (641) V: 11,301 (81) E: 8,497 (26)
US_FLAG (C4)	T: 19,943 (285) V: 8538(48) E: 6,447 (90)	T: 42052 (337) V: 10,970 (51) E: 8,497 (92)
MAPS (C5)	T: 19,943 (423) V: 8,538(161) E: 6,447 (142)	T: 41,988 (594) V: 11,290 (171) E: 8,473 (145)
MOUNTAIN (C6)	T: 19,943 (139) V: 8,538(154) E: 6,447 (65)	T: 42,073 (385) V: 11,331 (168) E: 8,496 (73)
PEOPLE_MARCHING (C7)	T: 19,943 (715) V: 8,538(209) E: 6,447 (86)	T: 42,021 (996) V: 11,321 (221) E: 8,473 (91)
PRISONER (C8)	T: 19,943 (43) V: 8,538(41) E: 6,447 (2)	T: 42,003 (61) V: 11,332 (43) E: 8,112 (2)
SPORTS (C9)	T: 19,943 (332) V: 8,538(240) E: 6,447 (98)	T: 41,753 (1,140) V: 11,310 (295) E: 8,498 (135)
WATERSCAPE_WATERFRONT (C10)	T: 19,943 (372) V: 8,538(122) E: 6,447 (92)	T: 42,043 (819) V: 11,312 (152) E: 8,484 (110)

V. RESULTS AND ANALYSIS

In this section the proposed *KernelRank* algorithm is analyzed using the development set for evaluating high-level feature extraction task in TRECVID 2005.

A. Experimental Setup

The development set consists of 74,509 keyframes that are extracted from 137 news videos (~80 hours). There are ten concepts used in NIST TRECVID'05 official evaluation. The news videos are in three languages, i.e. English, Chinese, and Arabic. Automatic speech recognition (*ASR*) technology is used to transcribe the audio channel into text, and machine translation (*MT*) technology is used to automatically translate Chinese and Arabic text into English text. All transcripts are provided at the shot level. Thus, there are two modality features, i.e. textual and visual, available for building the detection system. There are some shots in which the textual feature is absent due to various reasons such as recognition errors of ASR/MT or the audio channel having no speech. These shots are removed in our experiments based on the textual feature.

We randomly split the development set into the training set

(T), the validation set (V) and the evaluation set (E). The details, i.e. data size and positive sample number, of each data set are summarized in Table 1. The first column (*Column Concept*) lists the concept names together with their concept identities (bracket by parentheses); the second (*Column Textual*) describes the dataset for the textual features while the third (*Column Visual*) is for the visual feature. The concept *Building* (second row) and the textual feature (second column) are used to show how to read the numbers. In this example, the concept has 19,943 shots in the training set T, of which 2,008 shots are labeled as positive. It is abbreviated to 19,943 (2,008) in the table.

To collect enough information to classify the shot using the textual feature, we extract the 3,464-dimensional *tf-idf* feature from the text contained in the current shot and its 3 neighboring shots. To represent the visual content of image, we uniformly segment a keyframe into 77 grids, each having 32x32 pixels, from which a 12-dimensional texture feature (energy of log Gabor filter) is extracted. The visual feature is simple; however, our concern in this paper is to evaluate the efficiency of the learning algorithm for ranking rather than to select the best visual feature for representation.

B. Baseline Classifiers

The classifiers are LDF for the textual feature and 4-mixture GMM (mixture number is fixed for all concepts) for the visual feature. Because of the high cost of computation to fine-tune the mixture number of GMM for all concepts on the large dataset, the mixture number is empirically set to be 4. For the LDF classifier, a D-dimensional vector, X, is extracted for representing the sample. The ranking function is,

$$f\left(X;\Lambda\right)=W^{T}\cdot X \qquad (19)$$

with W being a D-dimensional parameter vector, i.e. Λ.

For the GMM classifier, we assume a set of D-dimensional vectors, X, is extracted from an image, and is denoted as $X=\left(\mathrm{x}_1,\mathrm{x}_2,\cdots,\mathrm{x}_\mathrm{L}\right)$ with $\mathrm{x}_\mathrm{i}\in R^{D}$ and L being the size of X. Then the ranking function is:

$$f\left(X;\Lambda\right)=\frac{1}{L}\left(\sum_{i=1}^{L}\log g^{+}\left(\mathrm{x}_\mathrm{i}\left|w^{+},\mu^{+},\Sigma^{+}\right.\right)-\sum_{i=1}^{L}\log g^{-}\left(\mathrm{x}_\mathrm{i}\left|w^{-},\mu^{-},\Sigma^{-}\right.\right)\right) \qquad (20)$$

where,

$$g^{c}\left(\mathrm{x}_\mathrm{i}\left|w^{c},\mu^{c},\Sigma^{c}\right.\right)=\sum_{k=1}^{K}w_k^{c}\cdot N\left(\mathrm{x}_\mathrm{i}\left|\mu_k^{c},\Sigma_k^{c}\right.\right),c=\left\{+,-\right\} \qquad (21)$$

with K being the mixture number, w_k^c the weights, and *N(.)*a Gaussian distribution with the mean μ_k^c and covariance matrix Σ_k^c (a diagonal matrix is used in our experiments). Thus, the parameter set is $\Lambda=\left\{w_k^c,\mu_k^c,\Sigma_k^c\right\}$, $k\in\left[1,K\right]$ and $c\in\left\{+,-\right\}$. Eq. (20) is the average likelihood ratio.

The GMM model is estimated using the EM algorithm. To initialize the starting points at the iterative EM, we first use the hierarchical k-means clustering algorithm to get the weights, means and variances of K clusters. The hierarchical clustering works as follows: first, one-mixture GMM is estimated. Then its center is split into two to obtain initial parameters of two-mixture GMM, from which two-mixture GMM is trained using the standard k-means algorithm. The above procedure runs until K-mixture GMM is reached. At each split, one cluster is increased. Finally, the EM algorithm starts to run from the estimated parameters to refine the K-mixture GMM model.

For each concept, the GMM (visual feature) /LDF (textual feature) classifiers are trained using the *KernelRank* algorithm for the Gaussian kernel and sigmoid-like kernel. For comparison, the following benchmark systems are built: (1) the classifier trained for maximizing AUC using the sigmoid function [23] (hereafter, it is named *sigmoid smoothing*); (2) the GMM classifier (for the visual feature) estimated using the EM algorithm introduced above; and (3) the SVM classifiers (for the textual feature) trained for minimizing classification error using the SVM^{light} [1] tool and for maximizing AUC using the SVM^{perf} [2]. To be a fair comparison, the linear kernel based SVM is trained to compare with the *KernelRank* based LDF. The SVM classifiers are trained because it is widely used for semantic concept detection in TRECVID, especially for SVM trained for classification error minimization.

C. Tuning Systems

The SVM^{light} and SVM^{Perf} tool packages should be carefully tuned for getting good results. The most important parameters to be adjusted are the kernel type, the trade-off coefficient between the training error and the margin, and the parameters of the corresponding kernel (e.g. it is the polynomial order for the polynomial kernel and the gamma for the RBF kernel). For the linear kernel, we only tune the trade-off coefficient. The optimal coefficient, which gives the best AUC value on the validation set, is determined through a 10-point grid search in a predefined range. The range is experimentally determined by 0.25 and 16 times of the default value (in SVM^{light}, the default value is calculated as the inverse of the average dot-product among training samples; in SVM^{Perf}, we set it equal to 20). In the 10-fold evaluation, we tune the configuration parameters based on one fold. And in large-scale experiments, the validation set is used for tuning the system. The results are reported based on the models trained using the best configuration.

In *KernelRank*, the parameters to be tuned are the variance σ for the Gaussian kernel and α for the sigmoid-like kernel and sigmoid function. The default value of α is set as the inverse of absolute average value of the scores calculated using the initial model on training samples. The default value of σ is set as the standard deviation of difference scores of the pair-wise samples on the training set. Similar to the tuning of SVM, the configuration is searched in a range of 0.1 and 10 times of the default value. The optimal value is the one that gives the maximal AUC value on the dataset used for tuning.

D. Evaluation Metrics

In addition to the AUC metric reported for comparing the systems, the average precision (*AP*), an official evaluation metric in TRECVID, is also given. The AP is defined as:

[1] http://www.cs.cornell.edu/People/tj/svm_light/index.html
[2] http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html

$$AP = \frac{1}{M}\sum_{i=1}^{Q}\frac{R_i}{i} * I_i \tag{22}$$

M is the number of true relevant (or positive) samples in the evaluation set. Q is the number of retrieved samples by the system, for example, Q=100 when only returning the top 100 samples for a given query. I_i is the *i-th* indicator in the rank list, which is equal to 1 if the *i-th* sample is relevant and zero otherwise. R_i is the number of relevant samples in the top i.

The AP value approximates the area under the Precision-Recall (*PR*) curve, which is an alternative view of the ROC curve. Thus the AP should be closely related with the AUC. We will experimentally show that the classifier trained for maximizing AUC will benefit the *AP* metric with the high chances.

E. Analysis on Small Dataset

First we analyze the proposed learning algorithm on a small dataset. The dataset is built by selecting 100 positive shots and 500 negative ones from the training set for the textual feature. The means of AUC values are reported as well as the mean of AP values on the 10-fold cross validation for 10 concepts.

The mean of AUC values and the corresponding standard deviation on the 10 folds are illustrated in Table 2 (Column *GAUS*: *KernelRank* with Gaussian kernel, Column *SIGL*: *KernelRank* with sigmoid-like kernel, Column *SIG*: sigmoid smoothing). The last row (Row: *Avg.*) is the average AUC values on ten concepts. For comparison, the SVM results are listed in the last two columns in the table (Column *SVM_C*: SVM for minimizing classification error, Column *SVM_AUC*: SVM for maximizing AUC).

First, the effects of two kernels discussed in Section III on the ranking measure are studied. As shown in the two left columns in Table 2, the *KernelRank* with the Gaussian gives the average AUC value of 94.5% vs. 93.9% for that with the sigmoid-like. They are comparable with the sigmoid smoothing (Column *SIG*). Second, *KernelRank* performs a little better than SVMs (Column *SVM_C* and *SVM_AUC*). Finally, comparison between SVM_C and SVM_AUC shows that SVM_C with an AUC value of 92.5% is competitive with the SVM_AUC with an AUC value of 92.2%.

To have an expressive analysis of different systems, the error bar is plotted in Figure 3 for visualizing the significance test (at the 95% confidence interval) based on the 10-fold AUC values. The X-axis is the concept identity while the Y-axis is the AUC value plus their error bars. For each concept, the columns from the left to the right are those for *KernelRank* with the Gaussian, sigmoid-like, sigmoid smoothing, SVM_C and SVM_AUC. It is obvious that although *KernelRank* works better than others for most of concepts, the improvement is not significant at the 95% confidence level.

As discussed above, the AP is another view of ranking performance. The system having better AUC should also have better AP values. To experimentally evaluate this property, the means of AP values on 10 folds are shown in Table 3. The AP value is calculated on the 10-fold evaluation. Similar observations to the above AUC analysis are obtained. Again, the *KernelRank* with the Gaussian ranks best among them with an average AP value of 83.4%, while *KernelRank* with the sigmoid-like kernel has a comparable AP value with the sigmoid smoothing. SVM_C is a little better than SVM_AUC. From the result of the pair-wise comparison between the AP values in Table 3 and the AUC values in Table 2, in general, the system with a better AUC correspondingly has a better AP value.

TABLE 2

MEAN OF AUC (%) VALUES PLUS STANDARD DEVIATION OF *KERNELRANK* FOR GAUSSIAN KERNEL (COLUMN GAUS) AND SIGMOID-LIKE KERNEL (COLUMN SIGL), SIGMOID SMOOTHING (COLUMN SIG), SVM FOR MINIMIZING CLASSIFICATION ERROR (SVM-C) AND SVM FOR MAXIMIZING AUC (SVM_AUC) (TEXTUAL)

CONCEPT	GAUS	SIGL	SIG	SVM_AUC	SVM_C
C1	87.5±7.9	85.7±8.4	86.0±7.9	83.2±9.2	83.7±8.2
C2	93.1±4.6	93.2±4.6	93.1±4.6	90.3±6.0	90.6±6.3
C3	95.2±3.1	94.9±2.9	94.7±3.1	94.8±2.8	93.4±3.4
C4	95.5±3.2	94.3±2.9	94.1±3.0	91.1±4.7	91.4±5.1
C5	93.3±5.3	93.1±5.5	93.1±5.6	89.1±6.7	90.7±4.9
C6	95.2±4.4	95.3±3.5	95.0±3.6	93.3±5.9	93.9±6.0
C7	94.0±5.1	93.8±5.2	93.9±5.3	92.7±5.8	93.4±5.2
C8	97.0±3.2	95.3±4.7	94.8±5.1	96.3±4.4	96.5±4.3
C9	99.6±0.5	99.3±0.9	99.4±1.0	98.6±2.0	98.5±2.0
C10	94.2±4.4	94.4±4.2	95.0±3.6	92.7±5.7	92.8±4.8
AVG.	94.5	93.9	93.9	92.2	92.5

TABLE 3

MEAN OF AP (%) VALUES PLUS STANDARD DEVIATION OF *KERNELRANK* FOR THE GAUSSIAN KERNEL (COLUMN GAUS) AND SIGMOID-LIKE KERNEL (COLUMN SIGL), SIGMOID SMOOTHING (COLUMN SIG), SVM FOR MINIMIZING CLASSIFICATION ERROR (SVM_C) AND SVM FOR MAXIMIZING AUC (SVM_AUC) (TEXTUAL)

CONCEPT	GAUS	SIGL	SIG	SVM_AUC	SVM_C
C1	71.4±13.5	70.5±13.3	70.4±12.7	69.5±12.0	69.4±13.3
C2	79.2±13.2	81.5±10.9	81.0±11.2	76.8±10.0	77.4±10.6
C3	81.1±11.8	80.4±11.5	79.8±12.1	80.6±10.9	79.4±10.9
C4	85.5±8.4	81.9±7.0	81.5±7.0	80.3±7.7	81.1±8.6
C5	80.5±10.2	79.9±8.7	79.1±10.1	74.2±9.6	80.5±5.1
C6	87.3±8.7	86.4±7.0	85.1±8.7	83.5±8.3	84.1±8.4
C7	87.1±9.1	87.8±7.9	88.8±8.1	85.7±10.4	88.3±7.2
C8	82.6±15.9	77.6±21.2	77.5±21.6	87.1±13.3	87.8±13.2
C9	98.2±2.3	97.4±3.3	97.9±2.9	94.1±6.6	95.1±6.0
C10	80.7±12.2	83.2±11.0	85.1±7.4	82.6±10.9	83.1±9.0
AVG.	83.4	82.7	82.6	81.4	82.6

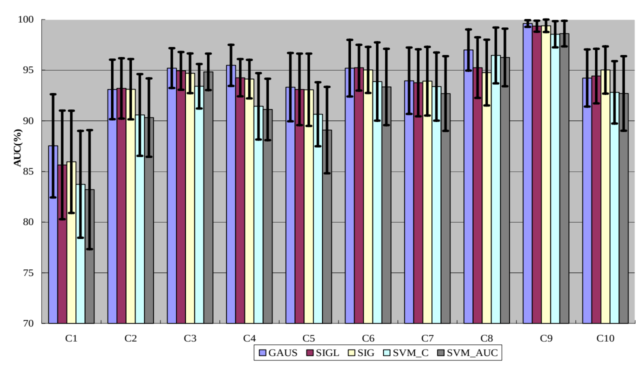


Fig. 3. Error bar illustration of AUC values (%) on 10 folds among the Gaussian kernel (GAUS), sigmoid-like kernel (SIGL), sigmoid smoothing (SIG), SVM classification (SVM_C), and ROC optimized SVM (SVM_AUC) (X-axis: concept identity, Y-axis: mean of AUC value plus 95% confidence interval).

F. Results on Large-scale Datasets

The second experiment is carried out on the large-scale dataset (details are shown in Table 1) for evaluating the ranking learning algorithms. Now the whole training set is used to learn the model parameters using the best configurations automatically determined based on the validation set. The AUC and AP values on the evaluation set are reported. In addition, the ROC curves for a few concepts are also given to visualize the behavior of learning algorithms at every operating point. The experiments are performed on the textual feature and visual feature independently. The AP value is calculated based on the full rank list of the test samples.

1) Textual feature

The AUC values on the evaluation set are shown in Table 4. The *KernelRank* results are in the columns *GAUS* for the Gaussian kernel and the column *SIGL* for the sigmoid-like kernel, respectively. The results of three benchmark systems are listed in the column *SIG* for sigmoid smoothing, *SVM_AUC* for SVM with maximizing AUC, and *SVM_C* for SVM with minimizing classification error.

The two types of kernels are comparable in terms of the overall average AUC and each concept. The average AUC values are 63.2% for the Gaussian kernel and 63.4% for the sigmoid-like one, respectively. Correspondingly, the system for the sigmoid smoothing has the average AUC value 63.7%. Thus, it is comparable with *KernelRank*. Second, we compare *KernelRank* with the SVM (Columns *SVM_AUC* and *SVM_C*). Obviously, *KernelRank* outperforms SVM models trained for maximizing AUC and for minimizing classification error. Finally, we analyze the two types of SVM models. As discussed on the small dataset (see Table 2), SVM_C is competitive with SVM_AUC. On the large-scale dataset, the SVM_AUC gets the average AUC value 61.8% on the evaluation set. It is significantly better than SVM_C, whose AUC value is only 55.7%. The reason may be that the data is much more highly unbalanced in the large dataset than the small dataset. For example, the ratio between the negative samples and the positive ones is about 5 in the small dataset for the 10-fold evaluation. However, it is increased to 16 in the training set for the concept *Building* (it is the concept having the smallest ratio on the training set among 10 concepts). More importantly, the ratio in the training set is much different from the evaluation set in the large-scale evaluation.

The corresponding AP values for the above experiments are shown in Table 5. Analyzing the AP values reaches the similar conclusions as analyzing the AUC. *KernelRank* ranks in the top performance in terms of AP values.

2) Visual feature

We demonstrate above the success of *KernelRank* on the textual feature by learning a simple LDF classifier. Now we move into the visual feature and learn a GMM classifier using the *KernelRank* for the Gaussian kernel and sigmoid-like kernel and the sigmoid smoothing. The benchmark system is trained by maximizing the likelihood.

Table 6 summaries the AUC values on ten concepts on the evaluation set. For each column in the table, from the left to the right, they are the AUC values for the *KernelRank* with the Gaussian (Column *GAUS*), *KernelRank* with the sigmoid-like (Column *SIGL*), sigmoid smoothing (Column *SIG*), and the benchmark (Column *ML*). Similar to the results on the textual feature, the performance of the two types of *KernelRank* systems are comparable. While compared with the sigmoid smoothing, no obvious improvement is observed. However, *KernelRank* performs significantly better than the benchmark system trained on the ML for all concepts. *KernelRank* has an AUC value 81.5% (sigmoid-like kernel) on the evaluation set. Correspondingly, the benchmark system only has 68.3%. It reminds us that learning the classifier for maximizing AUC is preferred for the ranking problem.

The AP values on the evaluation set are shown in Table 7. There is a little improvement on the AP values when comparing *KernelRank* with the sigmoid smoothing. They are 8.4% for the Gaussian kernel and 8.6% for the sigmoid-like kernel compared with 8.3% for the sigmoid smoothing. However, both of them are much better than 5.2%, the average AP value of the benchmark system.

TABLE 4
AUC (%) VALUES ON THE EVALUATION SET FOR THE GAUSSIAN KERNEL (COLUMN GAUS) AND SIGMOID-LIKE KERNEL (COLUMN SIGL), SIGMOID SMOOTHING (COLUMN SIG), SVM FOR MINIMIZING CLASSIFICATION ERROR (SVM_C) AND SVM FOR MAXIMIZING AUC (SVM_AUC) (TEXTUAL)

CONCEPT	GAUS	SIGL	SIG	SVM_AUC	SVM_C
C1	55.0	54.9	54.9	54.7	51.3
C2	66.3	66.5	66.0	66.2	62.4
C3	73.4	73.6	73.9	73.8	66.3
C4	71.0	70.7	71.0	71.3	58.0
C5	72.7	72.3	72.8	71.5	65.5
C6	65.9	67.5	67.1	61.7	53.7
C7	57.6	57.9	61.2	55.2	47.6
C8	27.5	27.6	27.6	25.3	26.6
C9	79.6	79.8	79.6	76.9	71.6
C10	63.2	63.2	63.2	60.9	54.3
AVG	63.2	63.4	63.7	61.8	55.7

TABLE 5
AP (%) VALUES ON THE EVALUATION SET FOR THE GAUSSIAN KERNEL (COLUMN GAUS) AND SIGMOID-LIKE KERNEL (COLUMN SIGL), SIGMOID SMOOTHING (COLUMN SIG), SVM FOR MINIMIZING CLASSIFICATION ERROR (SVM_C) AND SVM FOR MAXIMIZING AUC (SVM_AUC) (TEXTUAL)

CONCEPT	GAUS	SIGL	SIG	SVM_AUC	SVM_C
C1	18.0	18.0	18.0	17.7	15.1
C2	10.6	10.7	10.5	10.5	8.1
C3	2.2	2.2	2.2	3.1	3.1
C4	5.2	5.4	5.2	5.2	3.4
C5	11.8	11.8	11.7	11.7	8.2
C6	2.7	2.4	2.4	2.0	2.9
C7	1.7	1.7	2.0	1.6	1.3
C8	0.0	0.0	0.0	0.0	0.0
C9	18.7	17.7	18.7	17.7	21.0
C10	7.7	7.6	7.7	6.6	1.8
AVG	7.9	7.8	7.8	7.6	6.5

TABLE 6
AUC (%) VALUES ON THE EVALUATION SET FOR THE GAUSSIAN KERNEL (COLUMN GAUS) AND SIGMOID-LIKE KERNEL (COLUMN SIGL), SIGMOID SMOOTHING (COLUMN SIG) AND THE BENCHMARK (COLUMN ML) (VISUAL)

CONCEPT	GAUS	SIGL	SIG	ML
C1	70.0	69.7	69.7	62.0
C2	77.2	77.6	77.7	65.7
C3	82.0	82.3	82.6	74.9
C4	80.1	79.8	79.6	80.4
C5	84.9	83.9	83.9	76.9
C6	92.7	91.6	91.6	74.8
C7	81.3	85.3	85.1	80.5
C8	87.6	85.9	86.4	29.9
C9	78.5	78.7	76.8	72.0
C10	79.3	79.8	79.1	65.9
AVG	81.4	81.5	81.2	68.3

TABLE 7
AP (%) VALUES ON THE EVALUATION SET FOR THE GAUSSIAN KERNEL (COLUMN GAUS) AND SIGMOID-LIKE KERNEL (COLUMN SIGL), SIGMOID SMOOTHING (COLUMN SIG) AND THE BENCHMARK (COLUMN ML) (VISUAL)

CONCEPT	GAUS	SIGL	SIG	ML
C1	24.0	23.4	23.4	18.4
C2	14.4	14.4	14.5	8.4
C3	2.2	2.0	2.1	1.2
C4	5.4	5.4	5.4	3.1
C5	11.0	10.1	10.2	4.9
C6	9.0	9.6	9.8	3.7
C7	3.3	6.3	5.9	4.0
C8	0.2	0.1	0.1	0.0
C9	7.9	7.9	5.1	4.0
C10	6.9	7.0	7.0	3.9
AVG.	8.4	8.6	8.3	5.2

3) *Discussion*

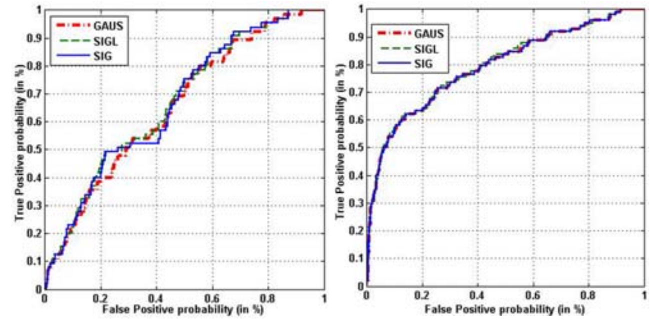
From the above experiments on the textual and visual features, we find that (1) *KernelRank* performs better than SVM classifiers trained for maximizing AUC or for minimizing classification error; (2) *KernelRank* outperforms the models trained for maximizing the likelihood; (3) for ranking problems, classifiers should be trained for maximizing AUC, e.g. SVM for maximizing AUC is better than SVM for minimizing classification error.

However, we cannot achieve significant gain from the kernel- based smoothing functions, at least for the current two types of functions, when compared with the sigmoid smoothing. The reason needs to be investigated in the future work. One possible reason may be that their experiential AUC curves (see Figure 1) are very similar to each other, although the kernel-based smoothing functions are different from the sigmoid smoothing. The good news is that *KernelRank* provides a way of designing and analyzing wide-scale smoothing functions based on the kernel. In future, we will further study the effect of other kernel families and develop a method of designing effective kernels based on the data.

4) *ROC curve analysis*

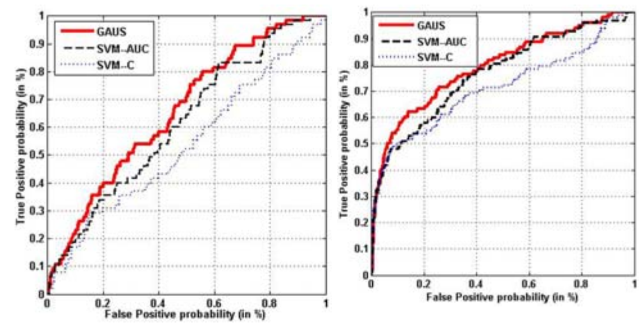
The AUC value is just a one-dimensional quality measure of the ROC curve (see discussions in Section II). Similarly, the AP is a one-dimensional quality measure of the PR curve. They only give an overall measure of a ranking system rather than the behavior at every operating point. The two-dimensional ROC curve shows more details.

In this section, the *KernelRank* algorithm is analyzed using the ROC curve. Due to limited space, we only select 2 concepts, i.e. *Mountain* and *Sports*, rather than 10 concepts for illustration, and draw their ROC curves based on the evaluation set.



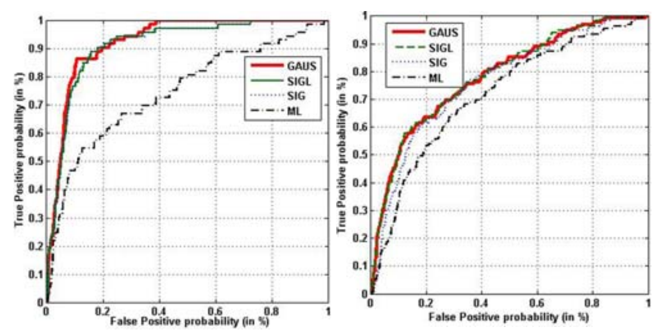
(a) C6 (Mountain, textual) (b) C9 (Sports, textual)

Fig.4. Compare ROC curves (textual feature) among KernelRank with Gaussian kernel (GAUS: red dash-dot line), KernelRank with sigmoid-like (SIGL: green dash line), and sigmoid smoothing (SIG: blue solid line) (X axis: false positive probability, Y axis: true positive probability)



(a) C6 (Mountain, textual) (b) C9 (Sports, textual)

Fig.5. Compare ROC curves (textual feature) among KernelRank with Gaussian kernel (GAUS: red solid line), AUC maximization based SVM (SVM_AUC: black dash line) and classification error minimization based SVM (SVM_C: blue dot line) (X axis: false positive probability, Y axis: true positive probability)



(a) C6 (Mountain, visual) (b) C9 (Sports, visual)

Fig.6. Compare ROC curves (visual feature) among KernelRank with the Gaussian (GAUS: red solid line), KernelRank with sigmoid-like (SIGL: green solid line), sigmoid smoothing (SIG: blue dot line), and the benchmark (black dash-dot line) (X axis: false positive probability, Y axis: true positive probability)

The ROC curves for the textual feature are plotted in Figures 4 and 5. Figure 4 visualizes the comparison between *KernelRank* (red dash-dot line for Gaussian kernel, *GAUS*, and green dash line for sigmoid-like kernel, *SIGL*) and sigmoid smoothing (blue thick line, *SIG*). At every decision point, *KernelRank* is comparable with the sigmoid-smoothing for the 2 concepts. It is also reflected by their corresponding AUC values (see Table 4), i.e. 65.9% (*GAUS*), 67.5% (*SIGL*) and 67.1% (*SIG*) for *Mountain* and 79.6% (*GAUS*), 79.8% (*SIGL*) and 79.6% (*SIG*) for *Sports*. Figure 5 compares the ROC curves between *KernelRank* with the Gaussian (red solid line, *GAUS*) and SVM (black dash line for AUC maximization based SVM, *SVM_AUC*, and blue dot line for classification error minimization based SVM, *SVM_C*). It is clearly seen that *KernelRank* outperforms AUC maximization based SVM at most of operating points, which is further better than the SVM for minimizing classification error. It is also indicated by their corresponding AUC values (see Table 4), i.e. 65.9% (*GAUS*) vs. 61.7% (*SVM_AUC*) and 53.7% (*SVM_C*) for *Mountain* and 79.6% (*GAUS*) vs. 76.9% (*SVM_AUC*) and 71.6% (*SVM_C*) for *Sports*. Moreover, the ROC reveals much more information than the AUC. For example, when the false positive probability is less than 0.1 in Figure 5, the true positive probabilities of the three systems are almost equal. This information is not provided by the one-scale AUC value.

The ROC curves for the same concept on the visual feature are depicted in Figure 6. Similarly, the ROC curves of *KernelRank* are close to those of the sigmoid smoothing. But at some operating points, a little improvement is observed. For example, the true positive probability reaches 1.0 (see Figure 6a) for Gaussian kernel when the false positive probability is between 0.4 and 0.6. As a comparison, the sigmoid smoothing has a true positive probability ~0.97.

Comparing *KernelRank* with the benchmark system trained for maximizing the likelihood (black dash-dot line, *ML*), an obvious improvement is obtained. At most of operating points, the AUC maximization based *KernelRank* outperforms the baseline. It is noted that even for the traditional classification problem, where the interested metric is accuracy, *KernelRank* is a better candidate rather than maximizing the likelihood or minimizing classification error using the MFoM [11, 12].

VI. CONCLUSION

An efficient kernel based learning algorithm, *KernelRank*, is presented for improving the performance of semantic concept detection. *KernelRank* designs a classifier for optimizing the ROC curve by directly maximizing a one-dimensional quality measure of the ROC curve, i.e. AUC, so that the parameters of classifiers are estimated. It exploits the kernel density estimation methods to model the ranking score distributions and to approximate the correcting ranking count. The model parameters are then naturally embedded into the objective function. To address the high cost of computation and memory in learning, an efficient implementation is developed based on the gradient descent algorithm. *KernelRank* is utilized to design two types of smoothing functions and to train the LDF and GMM classifiers. *KernelRank* is evaluated on the large-scale video dataset used for TRECVID 2005. Our experimental results and analysis show that (1) *KernelRank* is capable of training any differentiable classifier with various kernels; and (2) the learned ranking function performs better than the traditional maximization likelihood or classification error minimization based algorithms in terms of AUC and AP.

REFERENCES

[1] Bradley, A.P., The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30, pp. 1145-1159, 1997.

[2] Burges, C., et al. Learning to rank using gradient descent. *ICML'05*.

[3] Cortes, C. & Mohri, M. AUC optimization vs. error rate minimization. *NIPS'03*.

[4] Drummond, C. & Holte, R.C. What ROC curves can't do (and cost curves can). *ECAI'2004 workshop*.

[5] Duda, R. O., Hart, P.E. & Stock, D.GPattern classification. Wiley Interscience, 2nd edition, 2001.

[6] Egan, J. Signal detection theory and ROC analysis. *Academic Press*, 1975.

[7] Fawcett, T. ROC graphs: notes and practical considerations for researchers. *Technical Report HPL-2003-4*, HP Labs, 2003.

[8] Ferri, C., Flach, P. & Hernandez, J. Learning decision trees using the area under the ROC curve. *ICML'02*.

[9] Flach, P. A. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. *ICML'03*.

[10] Freund, Y., Iyer, R., Schapire, R. & Singer, Y. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4, 933-969, 2003.

[11] Gao, S., Wu W., Lee, C.-H. & Chua, T.-S. A MFoM learning approach to robust multiclass multi-label text categorization. *ICML'04*.

[12] Gao, S., Wu, W., Lee, C.-H. & Chua, T.-S. A maximal figure-of-merit approach to text categorization. *SIGIR'03*.

[13] Herschtal, A. & Raskutti, B. Optimising area under the ROC curve using gradient descent. *ICML'04*.

[14] Joachims, T. Learning to classify text using support vector machines. *Kluwer Academic Publishers*, 2002.

[15] Joachims, T., A support vector method for multivariate performance measures, *ICML'05*.

[16] Joachims, T., Optimizing Search Engines Using Clickthrough Data, *KDD'02*.

[17] Lehmann, E. L. *Testing Statistical Hypotheses*, John Wiley & Sons, 1959.

[18] Ling, X. & Yan, J. Decision tree with better ranking. *ICML'03*.

[19] Mozer, M. C., Dodier, R. & Colagrosso, M. Prodding the ROC curve: constrained optimization of classifier performance. *NIPS'02.*

[20] Prati, R. C. & Flach, P. A. ROCCER: A ROC convex hull rule learning algorithm. *ECML-PKDD'04 workshop, Advances in Inductive Rule Learning*.

[21] Rakotomamonjy, A. Support vector machines and area under ROC curves. *Technical report PSI-INSA de Rouen*, 2004.

[22] Rudin, C., Cortes, C., Mohri, M. & Schapire, R. Margin-based ranking meets boosting in the middle. *COLT'05*.

[23] Yan, L., Dodier R., Mozer, M.C. & Wolniewicz, R. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic, *ICML'03*.

[24] Gao, S. & Sun Q. B., Classifier optimization for multimedia semantic concept detection, I*CME'06*.

[25] Gao, S., Lee, C.-H. & Lim, J.-H., An ensemble classifier learning approach to ROC optimization, *ICPR'06*.

[26] Rudin, C., Ranking with a P-norm push, *COLT' 06*.

[27] Yang, Y.M. and Liu, X. A re-examination of text categorization methods. SIGIR'99.

Sheng Gao (M'04) received the B.E. in electronic engineering from North-western Polytechnical University, China in 1993, the M.E. in telecommunication from Beijing University of Posts and Telecommunications, China in 1996, and the Ph.D in pattern recognition and intelligent system from Institute of Automation, Chinese Academy of Sciences, China in 2001.

In 2001, he was Invited Researcher at the Spoken Language Translation Research Laboratories, Advanced Telecommunications Research Institute International, Japan, and in 2002, he was Research Fellow at the National University of Singapore. He is currently Senior Research Fellow at the Institute for Infocomm Research, Singapore. His research interests are focused on multimedia information retrieval, machine learning, speech recognition, and image and speech signal processing.

Qibin Sun received his Ph.D. degrees in Electrical Engineering, from University of Science and Technology of China (USTC), Anhui, China, in 1997. Since 1996, he is with the Institute for Infocomm Research, Singapore, where he is responsible for industrial as well as academic research projects in the area of media security, image and video analysis, etc. Dr. Sun is the Head of Delegates of Singapore in ISO/IEC SC29 WG1(JPEG). He worked in Columbia University during 2000-2001, as a research scientist.
Dr. Sun actively participates in professional activities such IEEE ICME, IEEE ISCAS, IEEE ICASSP and ACM MM, etc. He now serves as the member of Editorial Board in IEEE Multimedia Magazine, the member of Editorial Board in LNCS Transactions on Data Hiding and Multimedia Security, the member of Editorial Board in EUROSIP on Information Security and the Associate Editor of IEEE Transactions on Circuits and Systems on Video Technology