RankRC: Large-Scale Nonlinear Rare Class Ranking

Aditya Tayal, Thomas F. Coleman, and Yuying Li

Abstract—Rare class problems are common in real-world applications across a wide range of domains. Standard classification algorithms are known to perform poorly in these cases, since they focus on overall classification accuracy. In addition, we have seen a significant increase of data in recent years, resulting in many large scale rare class problems. In this paper, we focus on nonlinear kernel based classification methods expressed as a regularized loss minimization problem. We address the challenges associated with both rare class problems and large scale learning, by 1) optimizing area under curve of the receiver of operator characteristic in the training process, instead of classification accuracy and 2) using a rare class kernel representation to achieve an efficient time and space algorithm. We call the algorithm RankRC. We provide justifications for the rare class representation and experimentally illustrate the effectiveness of RankRC in test performance, computational complexity, and model robustness.

Index Terms—Machine learning, kernel-based learning, imbalanced classification, ranking loss, large-scale algorithms

1 Introduction

In many classification problems samples from one class are extremely rare (the minority class), while the number of samples belonging to the other class are plenty (the majority class). This situation is known as the rare class problem. It is also referred to as an unbalanced or skewed class distribution problem. Rare class problems naturally arise in several application domains, for example, fraud detection, customer churn, intrusion detection, fault detection, credit default, insurance risk and medical diagnosis.

Standard classification methods perform poorly when dealing with unbalanced data, e.g. support vector machines (SVM) [1], [2], [3], decision trees [1], [4], [5], [6], neural networks [1], Bayesian networks [7], and nearest neighbor methods [4], [8]. Most classification algorithms are driven by accuracy (i.e. minimizing error). Since minority examples constitute a small proportion of the data, they have little impact on accuracy or total error. Thus majority examples overshadow the minority class, resulting in models that are heavily biased in recognizing the majority class. Also, errors from different classes are assumed to have the same costs, which is usually not true. In most problems, incorrect classification of the rare class is more expensive, for instance, diagnosing a malignant tumor as benign has more severe consequences than the contrary case.

Solutions to the class imbalance problem have been proposed at both the data and algorithm level. At the data level, various resampling techniques are used to balance class distribution, including random under-sampling of majority class instances [9], over-sampling minority class instances with new synthetic data generation [10], and focused resampling, in which samples are chosen based on additional criteria [8]. Although sampling approaches have achieved success in some applications, they are known to have drawbacks, for instance under-sampling can eliminate useful information, while over-sampling can result in overfitting. At the algorithm level, solutions are proposed by adjusting the algorithm itself. This usually involves adjusting the costs of the classes to counter the class imbalance (cost-sensitive learning) or adjusting the decision threshold. However, true error costs are often unknown and using an inaccurate cost model can lead to additional bias.

In recent years, we have also seen an explosion of data, resulting in many large scale rare class problems. For example, detecting unauthorized use of a credit card from millions of transactions. Since frequently a nonlinear decision function is necessary to capture dependence structure, in this paper, we focus on nonlinear kernel based classification methods expressed as a regularized loss minimization problem. Processing large datasets can be prohibitive for many nonlinear kernel algorithms, which scale quadratically to cubically in the number of examples and may require quadratic space as well. In addition suitable loss function, which captures the imbalanced problem structure, needs to be selected.

To address the challenges associated with rare class problems and large scale learning, we propose the following:

1) Instead of maximizing accuracy (minimizing error), we optimize area under curve (AUC) of the receiver operator characteristic. The AUC overcomes inadequacies of accuracy for unbalanced problems and provides a skew independent measure. It is often used as the evaluation metric for unbalanced problems and therefore it is appropriate to directly optimize it in the training process. This results in a regularized biclass ranking problem, which is a special case of RankSVM with two ordinal levels [11].

- *A. Tayal and Y. Li are with the Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada. E-mail: {amtayal, yuying}@uwaterloo.ca.*
- *T.F. Coleman is with the Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. E-mail: tfcoleman@uwaterloo.ca.*

Manuscript received 10 June 2014; revised 4 Jan. 2015; accepted 29 June 2015. Date of publication 7 July 2015; date of current version 3 Nov. 2015. Recommended for acceptance by L. Khan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2453171

2) To solve a kernel RankSVM problem in the dual, as originally proposed in [11], requires $O(m^6)$ time and $O(m^4)$ space, where m is the number of data samples. Recently, [12] proposed a primal approach to solve RankSVM, which results in $O(m^3)$ time and $O(m^2)$ space for nonlinear kernels. We propose a modification to kernel RankSVM, that takes specific advantage of the unbalanced nature of the problem, using an iterative method for the matrix subproblem and assuming an upper bound on the number of iterations, to achieve $O(mm_+)$ time and $O(mm_+)$ space, where m_+ is the number of rare class examples. The idea is to restrict the solution to a linear combination of rare class kernel functions. We call it RankRC, since it enforces a Rare Class representation.

The main contributions of this paper are as follows:

- We focus on nonlinear (kernel) rare class learning based on AUC, which has not been highlighted in previous literature. Optimizing the AUC of the ROC curve corresponds to a binary RankSVM problem, which has been proposed in the literature previously by [11]. The proposed RankRC algorithm can be viewed as an approximation to kernel RankSVM. However, RankSVM has generally been used in the context of ranking (e.g., webpage ranking) with linear models. For rare class problems, predictive performance is improved by using the AUC as a loss function.
- We propose to use a rare class kernel representation to achieve significant improvement in computational complexity, while providing similar predictive performance as RankSVM. A limitation of using kernel RankSVM is computational complexity. Even with the primal approach proposed in [12], the method requires $O(m^3)$ complexity in time and $O(m^2)$ in space. This quickly becomes computationally impractical for large-scale rare class problems, where a balance ratio of 1:100 can result in sample sizes of millions of observations.
- The proposed rare class kernel presentation is a regularized kernel method that minimizes ranking loss. The rare class kernel representation idea is inspired by [13], in which the posterior probability density is estimated using a kernel density estimator over rare class samples and locally adjusted by the density of the background class. Using similar assumptions, we show the optimal solution can be approximately expressed as a linear combination of rare class kernel functions. Note, [13] does not propose a kernel method, but rather uses kernel density estimation to estimate the posterior probability density directly.
- We present extensive computational comparisons to demonstrate performance of the proposed RankRC using nonlinear kernels. Specifically, we demonstrate that, for rare class learning, RankSVM and RankRC, with AUC as an objective, generally perform better than error-rate based SVM methods. In addition, RankRC, which takes advantage of the typical density structure of the rare class problem, is computationally much more efficient than RankSVM, while not sacrificing performance effectiveness relative to RankSVM.

In this paper we concentrate on developing the rare class kernel model, the optimization algorithm, and extensive computational comparisons between AUC-based and error-rate based rare class nonlinear kernel learning, as well as computational efficiency improvement of RankRC over RankSVM. In addition to significant improvement in computational efficiency, we show that RankRC can also produce more robust out-of-sample models than RankSVM. We illustrate that limiting the flexibility of the nonlinear model to be expressed only as a combination of rare class kernel functions in RankRC is essentially performing a dimension reduction in the feature space. Using computational examples, we demonstrate that, for RankSVM, kernel and penalty parameters can more easily lead to overfitting the rare class samples. Since parameter selection (kernel and penalty) is an important component of using support-vector methods successfully, RankRC tends to provide more robust models as kernel and penalty parameters are evaluated.

The presentation of the paper is organized as follows. Sections 2 and 3 review the AUC measure and RankSVM. Section 4 develops the RankRC problem and presents justification for the rare class representation. Section 5 outlines the optimization method used to solve RankRC. Section 6 empirically compares RankRC with other kernel methods on several datasets. Finally, Section 7 concludes with summary remarks and potential extensions.

2 ROC Curve

Evaluation metrics play an important role in learning algorithms. They provide ways to assess performance as well as guide model learning. For classification problems, error rate is the most commonly used metric. For simplicity, we will consider the two-class case. Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$ be a set of m training examples, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y_i \in \{+1, -1\}$. Denote $f(\mathbf{x})$ as the inductive hypothesis obtained by training on example set $\mathcal{D}$. Then error rate is defined as,

$$\text{Error Rate} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[f(\mathbf{x}_i) \neq y_i], \tag{1}$$

where $\mathbb{I}[p]$ denotes the indicator function and is equal to 1 if p is true, 0 if p is false. However, for highly unbalanced datasets, error rate is not appropriate since it can be biased toward the majority class [14], [15], [16], [17]. In this paper, we follow convention and set the minority class as positive and the majority class as negative. Consider a dataset that has 1 percent positive cases and 99 percent negative ones. A naive solution which assigns every example to be positive will obtain only 1 percent error rate. Indeed, classifiers that always predict the majority class can obtain lower error rates than those that predict both classes equally well. But clearly these are not useful hypotheses.

Classification performance can be represented by a confusion matrix as in Table 1, with m_+ denoting the number of minority examples and m_- the number of majority ones. The proportion of the two rows reflects class distribution and any performance measure that uses values from both rows will be sensitive to class skew.

The receiver operating characteristic (ROC) can be used to obtain a skew independent measure [14], [18], [19]. Most

TABLE 1
Binary Classification Confusion Matrix

		Predicted		
		$f(\mathbf{x}) = +1$	$f(\mathbf{x}) = -1$	Total
Actual	$y = +1$	True Positives (TP)	False Positives (FP)	m_+
	$y = -1$	False Negatives (FN)	True Negatives (TN)	m_-

classifiers intrinsically output a numerical score and a predicted label is obtained by thresholding the score. For example, a threshold of zero leads to taking the sign of the numerical output as the label. Each threshold value generates a confusion matrix with different quantities of false positives and negatives. The ROC graph is obtained by plotting the true positive rate (number of true positives divided by m_+) against the false positive rate (number of false positives divided by m_-) as the threshold level is varied (see Fig. 1). It depicts the trade-off between benefits (true positive) and costs (false positives) for different choices of the threshold. Thus it does not depend on a priori knowledge of the costs associated with misclassification. A ROC curve that dominates another provides a better solution at any cost point.

To facilitate comparison, it is convenient to characterize ROC curves using a single measure. The area under a ROC curve (AUC) can be used for this purpose. It is the average performance of the model across all threshold levels and corresponds to the Wilcoxon rank statistic [20]. The AUC can be obtained by forming the ROC curve and using the trapezoid rule to compute area. Also, given the intrinsic output of a hypothesis, $f(\mathbf{x})$, we can directly compute the AUC by counting pairwise correct rankings [21]:

$$\mathrm{AUC} = \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \mathbb{I}\big[f(\mathbf{x}_i) > f(\mathbf{x}_j)\big] \,. \tag{2}$$

We recognize that potential inadequacies, e.g., including performance over the ROC space in which one rarely consider, can arise from using AUC as the performance measure, see, e.g., [22], [23]. However, assuming no additional problem dependent information such as error costs, AUC evaluates model performance based on bi-class criteria, instead of the single total error rate. Optimizing AUC often leads to a dominant performance ROC curve, achieving better performance with respect to both the true positive rate and false positive rate criteria, in the absence of the additional problem dependent information.

Incorporating the AUC in the modeling process leads to a biclass ranking problem, as discussed in the following section.

3 RankSVM

The modeling process can usually be expressed as an optimization problem involving a loss function and a penalty on complexity (e.g. regularization term). For most classification problems, since the performance measure is error rate, it is natural to consider minimizing the empirical error rate (1) as the loss function. In practice, $\mathbb{I}[\cdot]$ is often replaced with a convex approximation such as the hinge loss, logistic loss or exponential loss [24]. Specifically, using the hinge loss,

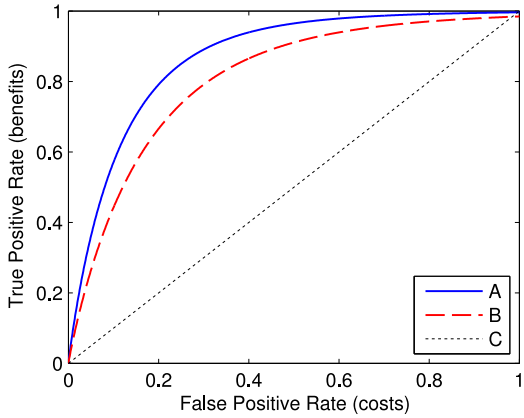


Fig. 1. Example ROC curves. Curve A dominates B and curve B dominates C. Curve C has an AUC of 0.5 and indicates a model with no discriminative value.

$\ell_h(z) = \max(0, 1 - z)$, with ℓ_2-regularization leads to the well known support vector machine formulation [25], [26],

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m} \ell_h\big(y_i \mathbf{w}^T \mathbf{x}_i\big) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,, \tag{3}$$

where $\lambda \in \mathbb{R}_+$ is a parameter that controls complexity and the hypothesis, $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, is assumed linear in the input space $\mathcal{X}$. Since SVMs try to minimize error rate, they can lead to ineffective class boundaries when dealing with highly skewed datasets, with resulting solutions biased toward the majority concept [3]. The literature contains several approaches to remedy this problem. Most prevalent are sampling methods and cost-sensitive learning. However, these approaches explicitly or implicitly fix the relative costs of misclassification. When the true costs are unknown, this can lead to suboptimal solutions.

Instead of minimizing error rate, we consider optimizing AUC as a natural way to deal with imbalance. Indeed, if we measure performance using AUC, it is preferable to optimize this quantity directly during the training process. In the AUC formula given in (2), we replace $\mathbb{I}[\cdot]$ with the hinge loss to obtain a convex ranking loss function. Thus we solve the following regularized loss minimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_h\big(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j\big) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,. \tag{4}$$

Problem (4) is a special case of RankSVM proposed by [11] with two ordinal levels. Like SVM, RankSVM leads to a dual problem which can be expressed in terms of dot-products between input vectors. This allows us to obtain a non-linear function through the kernel trick [25], which consists of using a kernel function, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, that corresponds to a feature map, $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$, such that $\forall \mathbf{u}, \mathbf{v} \in \mathcal{X}$, $k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T\phi(\mathbf{v})$. Here, k directly computes the inner product of two vectors in a potentially high-dimensional feature space $\mathcal{F}$, without the need to explicitly form the mapping. Consequently, we can replace all occurrences of the dot-product with k in the dual and work implicitly in space $\mathcal{F}$.

However, since there is a Lagrange multiplier for each constraint associated with the hinge loss, the dual formulation leads to a problem in $m_+m_- = O(m^2)$ variables. Assuming

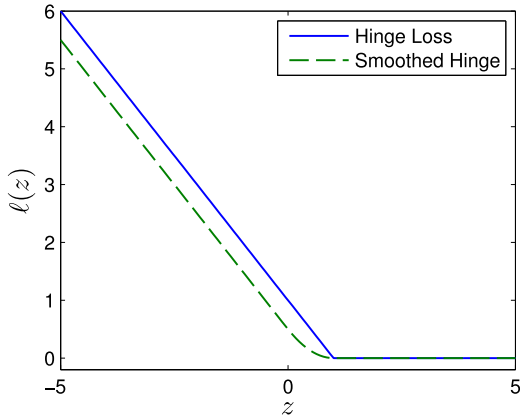


Fig. 2. The smoothed hinge is a differentiable approximation of the hinge loss. Here the smoothed hinge is shown with $\epsilon = 0.5$.

the optimization procedure has cubic complexity in the number of variables and quadratic space requirements, the complexity of the dual method is $O(m^6)$ time and $O(m^4)$ space, which is unreasonable for even medium sized datasets.

As noted by Chapelle and Keerthi in [12], [27], one can also solve the primal problem in the implicit feature space due to the Representer Theorem [28], [29]. This theorem states that the solution of any regularized loss minimization problem in $\mathcal{F}$ can be expressed as a linear combination of kernel functions evaluated at the training samples, $k(\mathbf{x}_i, \cdot)$, $i = 1, \ldots, m$. Thus, the solution of (4) in $\mathcal{F}$ can be written as:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \mathbf{x}) \ , \text{ or } \mathbf{w} = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \cdot) \ . \tag{5}$$

Substituting (5) in (4) we can express the primal problem in terms of $\boldsymbol{\beta}$:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^m} \quad \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_h \left(\sum_{r=1}^{m} \beta_r k(\mathbf{x}_r, \mathbf{x}_i) - \sum_{r=1}^{m} \beta_r k(\mathbf{x}_r, \mathbf{x}_j) \right) + \frac{\lambda}{2} \sum_{i,j=1}^{m} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j) \ ,$$

or more simply,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^m} \quad \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_h \big(K_i \boldsymbol{\beta} - K_j \boldsymbol{\beta} \big) + \frac{\lambda}{2} \boldsymbol{\beta}^T K \boldsymbol{\beta} \ , \tag{6}$$

where $K \in \mathbb{R}^{m \times m}$ is the kernel matrix, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and K_i denotes the ith row of K. To be able to solve (6) using unconstrained optimization methods such as gradient descent, we require the objective to be differentiable.

We replace the hinge loss, ℓ_h, with an ϵ-smoothed differentiable approximation, ℓ_ϵ, defined as,

$$\ell_\epsilon(z) = \begin{cases} (1-\epsilon) - z & \text{if } z < 1 - 2\epsilon \\ \frac{1}{4\epsilon}(1-z)^2 & \text{if } 1 - 2\epsilon \leq z < 1 \\ 0 & \text{if } z \geq 1, \end{cases}$$

which transitions from linear cost to zero cost using a quadratic segment (see Fig. 2) and provides similar benefits as the hinge loss. Thus we can solve (6) using standard unconstrained optimization techniques. Since there are m variables, Newton's method would for example take $O(m^3)$ operations to converge.

RankSVM is popular in the information retrieval community, where linear models are the norm [30]. Computational implementations for linear models have been considered, see, e.g., [31]. For a linear model, with d-dimension input vectors, the complexity of RankSVM can be reduced to $O(md + m \log\ m)$ [12]. However, many rare class problems require a nonlinear function to achieve optimal results. Solving a nonlinear RankSVM requires $O(m^3)$ time and $O(m^2)$ space [12], [32], which is not practical for mid-to large-sized datasets. We believe this complexity is, in part, the reason why nonlinear RankSVMs are not commonly used to solve rare class problems.

In the next section we propose a modification to nonlinear RankSVMs that takes specific advantage of unbalanced datasets to achieve $O(mm_+)$ time and $O(mm_+)$ space, while not sacrificing performance.

4 RankRC: Ranking with Rare Class Representation

To make RankSVM computationally feasible for large scale unbalanced problems, we propose to enforce a rare class representation for the decision surface. Specifically, we propose to restrict the solution to the form

$$f(\mathbf{x}) = \sum_{y_i=+1} \beta_i k(\mathbf{x}_i, \mathbf{x}) \ , \tag{7}$$

so it consists only of kernel function realizations of the minority class. We call this RankRC to indicate a Rare Class representation, instead of a support vector representation.

We present motivation for RankRC by assuming specific properties of the class conditional distributions and kernel function. Similar assumptions are made in [13]; however likelihood ratio is directly estimated in [13] instead. In contrast, we are using a regularized loss minimization approach.

Recall that the optimal ranking function for a classification problem is the posterior probability, $P(y = 1|\mathbf{x})$, since it minimizes the Bayes risk for arbitrary costs. From Bayes' Theorem, we have

$$P(y=1|\mathbf{x}) = \frac{P(y=1)P(\mathbf{x}|y=1)}{P(y=1)P(\mathbf{x}|y=1) + P(y=-1)P(\mathbf{x}|y=-1)} \ . \tag{8}$$

Any monotonic transformation of (8) also yields equivalent ranking capability. Let $f(\mathbf{x})$ denote the likelihood ratio, i.e.,

$$f(\mathbf{x}) = \frac{P(\mathbf{x}|y=1)}{P(\mathbf{x}|y=-1)}. \tag{9}$$

Dividing the numerator and denominator of (8) by $P(y=-1)P(\mathbf{x}|y=-1)$, we have

$$P(y=1|\mathbf{x}) = \frac{\rho f(\mathbf{x})}{\rho f(\mathbf{x}) + 1}, \quad \text{where} \quad \rho = \frac{P(\mathbf{x}|y=1)}{P(\mathbf{x}|y=-1)} > 0.$$

We note that $P(y = 1|\mathbf{x})$ is a monotonic transformation of the likelihood ratio, which is the ranking function we are interested in obtaining. If we assume that the conditional

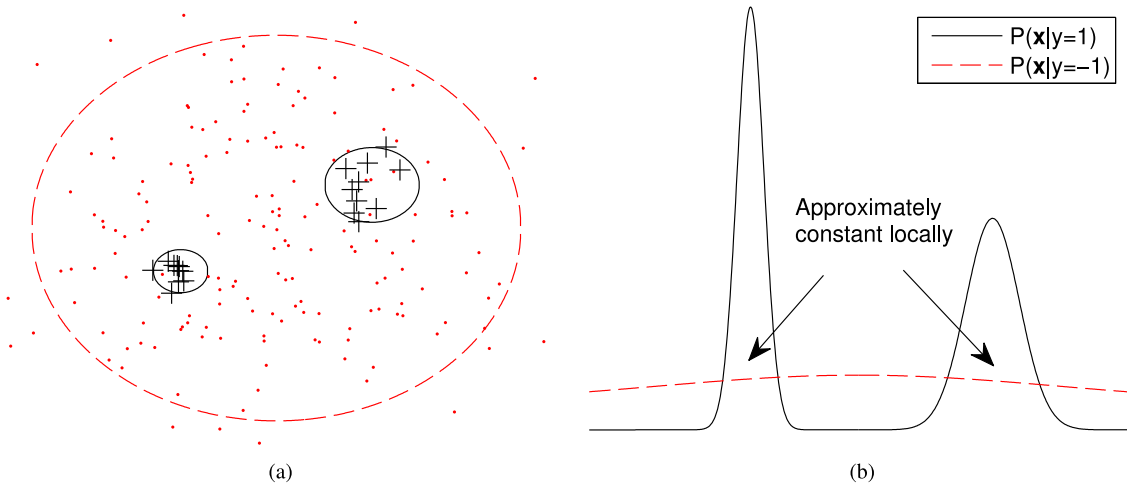


Fig. 3. (a) An example of a rare class dataset. Red '·'s indicate negative (majority) examples and black '+'s indicate positive (minority) examples. (b) The class conditional distributions showing that $P(\mathbf{x}|y=-1)$ is relatively constant in local neighborhood of positive examples.

density, $P(\mathbf{x}|y=1)$, is a mixture of m_+ identical spherical normals centered at the rare class examples, we can write

$$P(\mathbf{x}|y=1) = \sum_{y_i=+1} a_i \exp\left\{\frac{||\mathbf{x}_i - \mathbf{x}||^2}{\sigma^2}\right\},$$

for some constants a_i. This mixture encompasses a large range of possible distributions from the m_+ rare examples provided. If we also assume that k denotes a Gaussian kernel function with width σ, then we have

$$P(\mathbf{x}|y=1) = \sum_{y_i=+1} a_i k(\mathbf{x}_i, \mathbf{x}) \ . \tag{10}$$

In rare class problems, most examples are from the majority class ($y=-1$) and only a small number are from the rare class ($y=1$). It is reasonable to assume the minority class examples are concentrated in local regions with bounded support, while the majority class acts as background noise. This assumption is valid for rare class problems involving statistical detection. In unbalanced problems, in which the objective is outlier detection, these assumptions may not hold as strongly. For example, tumor detection or patient hospitalization requires statistical detection to identify the relevant patterns in noisy data, whereas identifying unusual credit card purchasing behavior may be better to suited as an outlier problem. We note that [13] make use of similar assumptions in their method to directly estimate the likelihood ratio and apply their method successful to a real drug discovery dataset.

Therefore, in a neighborhood around the minority class examples, the conditional density function $P(\mathbf{x}|y=-1)$ can be assumed to be relatively flat in comparison to $P(\mathbf{x}|y=1)$, see Fig. 3 for example. Let $P(\mathbf{x}|y=-1) \approx c_i$ for each minority example i in the neighborhood of $\mathbf{x}_i$.[1] Then together with (10), the likelihood ratio (9) can be written as

$$f(\mathbf{x}) \approx \sum_{y_i=+1} \frac{a_i k(\mathbf{x}_i, \mathbf{x})}{c_i} \ ,$$

1. We do not make this more precise since we are mainly interested in motivating an approximate form.

which corresponds to the rare class representation (7) we have chosen. In contrast to (5), this formulation takes specific advantage of the conditional density structure of rare class problems.

The above analysis suggests that, for rare class ranking problems, the restricted hypothesis function in (7) is a reasonable approximation to the full data hypothesis function $f(\mathbf{x}) = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \mathbf{x})$. More generally, we consider solving the regularized ranking problem under an *arbitrary* subset representation $\mathcal{R}$,

$$\begin{aligned} \min_{\boldsymbol{\beta}\in\mathbb{R}^{|\mathcal{R}|}} \quad & \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_h\big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big) + \frac{\lambda}{2} \boldsymbol{\beta}^T K_{\mathcal{R}\mathcal{R}} \boldsymbol{\beta} \\ \text{s.t.} \quad & f(\mathbf{x}) = \sum_{r\in\mathcal{R}} \beta_r k(\mathbf{x}_r, \mathbf{x}), \end{aligned} \tag{11}$$

where $\mathcal{R} \subseteq \{1, \ldots, m\}$ denotes the subset and $K_{\mathcal{R}\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{R}|}$ corresponds to the square kernel submatrix indexed by $\mathcal{R}$. When $\mathcal{R}$ is the full training set, (11) corresponds to RankSVM, while RankRC corresponds to setting $\mathcal{R} = \{r : y_r = 1\}$.

Recall that $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$ denote a feature map corresponding to the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that $\forall \mathbf{u}, \mathbf{v} \in \mathcal{X}$, $k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T \phi(\mathbf{v})$. Let $\mathcal{S}_\mathcal{R} = \text{span}\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\}$ be a linear combination of the subset of points in feature space indexed by $\mathcal{R}$. Let $\phi_\mathcal{R} : \mathcal{X} \to \mathcal{F}_\mathcal{R} \subseteq \mathbb{R}^{d'}$, defined as the orthogonal projection of ϕ onto $\mathcal{S}_\mathcal{R}$, i.e.,

$$\phi_\mathcal{R}(\mathbf{x}) = \text{Proj}_{\mathcal{S}_\mathcal{R}}(\phi(\mathbf{x})) \ . \tag{12}$$

It can be readily observed that

$$\begin{aligned} & \sum_{i\in\mathcal{R}} \beta_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \\ &= \left(\sum_{i\in\mathcal{R}} \beta_i \phi(\mathbf{x}_i), \phi(\mathbf{x}) \right) \\ &= \left(\text{Proj}_{\mathcal{S}_\mathcal{R}} \left(\sum_{i\in\mathcal{R}} \beta_i \phi(\mathbf{x}_i) \right), \text{Proj}_{\mathcal{S}_\mathcal{R}}(\phi(\mathbf{x})) \right) \\ &= \left(\sum_{i\in\mathcal{R}} \beta_i \phi_\mathcal{R}(\mathbf{x}_i), \phi_\mathcal{R}(\mathbf{x}) \right). \end{aligned}$$

In other words, using the restricted hypothesis is equivalent to using the projected mapping $\phi_{\mathcal{R}}(\cdot)$, instead of the original feature mapping $\phi(\cdot)$. If $\text{span}\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\}$ is a proper subspace of $\text{span}\{\phi(\mathbf{x}_i) : i = 1, \ldots, m\}$, we can view RankRC as first performing dimension reduction in the feature space based on the rare class samples, before applying regularized risk minimization. This is likely to lead to more robust learning; this will be further illustrated with computational examples in Section 6.

5 Computational Complexity Comparison between RankRC over RankSVM

Setting $\mathcal{R} = \{r : y_r = 1\}$ in (11) and replacing ℓ_h with the smooth approximation, ℓ_ϵ, we obtain the following RankRC problem in m_+ variables,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{m_+}} \quad \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_\epsilon\big(K_{i+}\boldsymbol{\beta} - K_{j+}\boldsymbol{\beta}\big) + \frac{\lambda}{2}\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta} \,. \tag{13}$$

Here, K_{i+} denotes ith row of K with column entries corresponding to only the positive class, and $K_{++} \in \mathbb{R}^{m_+ \times m_+}$ is the square submatrix of K corresponding to the positive class entries. To solve (13) we can use several approaches, which are discussed below.

5.1 Linearization

Since K_{++} is a positive semi-definite matrix, it has an eigen-decomposition which can be expressed in the form, $K_{++} = U\Lambda U^T$, with U being an orthonormal matrix (i.e. $U^T U = I$) and Λ a diagonal matrix containing non-negative eigenvalues of K_{++}. Let $\mathbf{w} = \Lambda^{\frac{1}{2}} U^T \boldsymbol{\beta}$, then

$$\boldsymbol{\beta} = U\Lambda^{\dagger\frac{1}{2}}\mathbf{w} \,, \tag{14}$$

where $\Lambda^\dagger$ denotes the pseudoinverse of Λ. We can substitute (14) in (13) to obtain the following linear (hypothesis) space problem,

$$\min_{\mathbf{w} \in \mathbb{R}^{m_+}} \quad \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_\epsilon\Big(K_{i+}U\Lambda^{\dagger\frac{1}{2}}\mathbf{w} - K_{j+}U\Lambda^{\dagger\frac{1}{2}}\mathbf{w}\Big) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,. \tag{15}$$

That is, Problem (15) is equivalent to Problem (4) with data points given by $\mathbf{x}_i = (K_{i+}U\Lambda^{\dagger\frac{1}{2}})^T = \Lambda^{\dagger\frac{1}{2}}U^T K_{i+}^T \in \mathbb{R}^{m_+}$, $i = 1, \ldots, m$. Therefore we can use the algorithm described in [12] to solve the linear ranking problem in $O(mm_+ + m \log\, m) = O(mm_+)$ time. The cost of computing $\mathbf{x}_i = K_{i+}U\Lambda^{\dagger\frac{1}{2}}, i = 1, \ldots, m$, is $O(mm_+^2)$. The cost of factoring K_{++} is $O(m_+^3)$. Therefore the total time is $O(mm_+^2 + m_+^3)$. Once we solve for optimal $\mathbf{w}$ we can use (14) to obtain $\boldsymbol{\beta}$ for subsequent testing purposes. Also, since we only need kernel entries $\{K_{ij} : y_i = 1, j = 1, \ldots, m\}$, the method uses $O(mm_+)$ space.

5.2 Unconstrained Optimization

We can also directly solve (13) using standard unconstrained optimization methods. Gradient only methods, such as steepest descent and nonlinear conjugate gradient do not require estimation of the Hessian. Although this makes each iteration much cheaper, convergence can be slow, especially near the solution. In contrast Hessian based algorithms, such as Newton's method can obtain quadratic convergence near the solution, but each iteration can be expensive. In Newton's method, the pth iterate is updated according to

$$\boldsymbol{\beta}^{(p+1)} = \boldsymbol{\beta}^{(p)} + \mathbf{s} \,,$$

where the step, $\mathbf{s}$, is obtained by minimizing the quadratic Taylor approximation around the current iterate $\boldsymbol{\beta}^{(p)}$:

$$\min_{\mathbf{s}} \quad \mathbf{s}^T\mathbf{g}^{(p)} + \frac{1}{2}\mathbf{s}^T H^{(p)}\mathbf{s} \,, \tag{16}$$

where $H^{(p)}$ and $\mathbf{g}^{(p)}$ are the Hessian and gradient of the objective at $\boldsymbol{\beta}^{(p)}$, respectively. Problem (16) has a closed form solution given by

$$\mathbf{s} = -(H^{(p)})^{-1}\mathbf{g}^{(p)} \,.$$

Since $H^{(p)}$ is a $m_+ \times m_+$ matrix, this involves $O(m_+^3)$ cost in each iteration. To avoid this, we can use the truncated Newton method in which $H^{(p)}\mathbf{s} = -\mathbf{g}^{(p)}$ is solved using linear conjugate gradient. Here, the Hessian is not computed explicitly and the method iteratively approximates the solution using Hessian-vector products. Since each iteration in the linear conjugate gradient algorithm leads to a descent direction, we can terminate early while still improving convergence.

A drawback of (truncated) Newton's method is that it is locally convergent. If the initial point is not chosen close enough to the solution, the method can be slow to converge, or fail altogether. Therefore we consider a subspace-trust-region method, which combines the benefit of a truncated Newton step with steepest descent. In our tests, we found that the subspace-trust-region method converges with significantly fewer iterations than the truncated Newton method.

The idea behind the trust-region method is to solve (16) while constraining the step, $\mathbf{s}$, to a neighborhood around the current iterate, in which the approximation is trusted:

$$\begin{aligned} \min_{\mathbf{s}} \quad & \frac{1}{2}\mathbf{s}^T H^{(p)}\mathbf{s} + \mathbf{s}^T\mathbf{g}^{(p)} \\ \text{s.t.} \quad & ||\mathbf{s}||_2 \leq \Delta^{(p)} \,. \end{aligned} \tag{17}$$

The trust region radius, $\Delta^{(p)}$, is adjusted at each iterate according to standard rules, for example it is decreased if the solution obtained is worse than the current iterate. Problem (17) can be solved accurately, see, e.g., [33], however, the solution uses the full eigen-decomposition of $H^{(p)}$. To avoid this computation, in the subspace-trust-region method, Problem (17) is restricted to a two-dimensional subspace spanned by the gradient, $\mathbf{g}^{(p)}$, and an approximate Newton direction, $\mathbf{s}_2$, which can be obtained by solving $H^{(p)}\mathbf{s}_2 = -\mathbf{g}^{(p)}$ using linear conjugate gradient [34]. The idea behind this choice is to ensure global convergence, while maintaining fast local convergence. Once the subspace has been computed, solving (17) costs $O(1)$ time, since in the subspace the problem is only two-dimensional.

The implementation we use is provided in Matlab's optimization toolbox, `fminunc/fmincon`.

5.2.1 Computing Gradient and Hessian-Vector Product

We describe how we can compute the gradient and Hessian-vector product for Problem (13) efficiently. Let $K_{\cdot+} = [K_{ij}]_{i=1,\ldots,m,y_j=1} \in \mathbb{R}^{m\times m_+}$ denote the rectangular submatrix of K with columns indexed by the positive class. Consider the expanded matrix

$$A = [K_{i+} - K_{j+}]_{i:y_i=1,j:y_j=-1} \in \mathbb{R}^{m_+m_-\times m_+} ,$$

consisting of the differences of rows in $K_{\cdot+}$ corresponding to all pairwise preferences. In our computation we do not explicitly form matrix A, rather we note that A can be expressed as a sparse matrix product:

$$A = PK_{\cdot+},$$

where $P \in \mathbb{R}^{m_+m_-\times m}$ is a sparse matrix that encodes a pairwise preference. That is, if $y_i > y_j$, then there exists a row r in P such that $P_{ri} = 1, P_{rj} = -1$ and the rest of the row is zero. Let A_r denote the rth row of A. Then the ranking loss expression in (13) can be written as,

$$\begin{aligned}
&\sum_{y_i=+1}\sum_{y_j=-1} \ell_\epsilon\big(K_{i+}\boldsymbol{\beta} - K_{j+}\boldsymbol{\beta}\big) \\
&\quad= \sum_{r=1}^{m_+m_-} \ell_\epsilon(A_r\boldsymbol{\beta}) \\
&\quad= \sum_{r=1}^{m_+m_-} \mathbb{I}[r\in\mathcal{L}](1-\epsilon-A_r\boldsymbol{\beta}) + \sum_{r=1}^{m_+m_-} \mathbb{I}[r\in\mathcal{Q}]\frac{1}{4\epsilon}(1-A_r\boldsymbol{\beta})^2 ,
\end{aligned} \tag{18}$$

where $\mathcal{L} = \{r : A_r\boldsymbol{\beta} < 1-2\epsilon\}$ is the set of pairwise differences which are in the linear portion of ℓ_ϵ, and $\mathcal{Q} = \{r : 1-2\epsilon \le A_r\boldsymbol{\beta} < 1\}$ is the set which fall in the quadratic part. Denote $\mathbf{e} \in \mathbb{R}^{m_+m_-}$ as a vector of ones. Define $\mathbf{e}^{\mathcal{L}} \in \mathbb{R}^{m_+m_-}$ as a binary vector where $\mathbf{e}^{\mathcal{L}}_r = 1$ if $r \in \mathcal{L}$ and $\mathbf{e}^{\mathcal{L}}_r = 0$ if $r \notin \mathcal{L}$. Also define $I^{\mathcal{Q}} \in \mathbb{R}^{m_+m_-\times m_+m_-}$ as a diagonal matrix, where $I^{\mathcal{Q}}_{rr} = 1$, if $r \in \mathcal{Q}$, and $I^{\mathcal{Q}}_{rr} = 0$, if $r \notin \mathcal{Q}$. Then (18) is equivalent to

$$\begin{aligned}
&\big(\mathbf{e}^{\mathcal{L}}\big)^T((1-\epsilon)\mathbf{e} - A\boldsymbol{\beta}) + \frac{1}{4\epsilon}(\mathbf{e}-A\boldsymbol{\beta})^T I^{\mathcal{Q}}(\mathbf{e}-A\boldsymbol{\beta}) \\
&= \big(\mathbf{e}^{\mathcal{L}}\big)^T((1-\epsilon)\mathbf{e} - PK_{\cdot+}\boldsymbol{\beta}) + \frac{1}{4\epsilon}(\mathbf{e}-PK_{\cdot+}\boldsymbol{\beta})^T I^{\mathcal{Q}}(\mathbf{e}-PK_{\cdot+}\boldsymbol{\beta}).
\end{aligned}$$

Therefore the objective function in (13) can be expressed as

$$\begin{aligned}
F(\boldsymbol{\beta}) \triangleq \frac{1}{m_+m_-}\Big[&\big(\mathbf{e}^{\mathcal{L}}\big)^T((1-\epsilon)\mathbf{e} - PK_{\cdot+}\boldsymbol{\beta}) \\
&+ \frac{1}{4\epsilon}(\mathbf{e}-PK_{\cdot+}\boldsymbol{\beta})^T I^{\mathcal{Q}}(\mathbf{e}-PK_{\cdot+}\boldsymbol{\beta})\Big] + \frac{\lambda}{2}\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta} .
\end{aligned} \tag{19}$$

We obtain the gradient by taking the derivative of (19) with respect to $\boldsymbol{\beta}$:

$$\begin{aligned}
\mathbf{g} &\triangleq \frac{\partial F}{\partial \boldsymbol{\beta}} \\
&= \frac{1}{m_+m_-}\Big[-\big(\mathbf{e}^{\mathcal{L}}\big)^T PK_{\cdot+} + \frac{1}{2\epsilon}PK_{\cdot+}I^{\mathcal{Q}}(PK_{\cdot+}\boldsymbol{\beta}-\mathbf{e})\Big] + \lambda K_{++}\boldsymbol{\beta} \\
&= \frac{1}{m_+m_-}\Big[-((\mathbf{e}^{\mathcal{L}})^T P)K_{\cdot+} + \frac{1}{2\epsilon}\big(P(K_{\cdot+}\big(I^{\mathcal{Q}}P\big)(K_{\cdot+}\boldsymbol{\beta})\big) \\
&\qquad - P\big(K_{\cdot+}\big(I^{\mathcal{Q}}\mathbf{e}\big)\big)\big)\Big] + \lambda K_{++}\boldsymbol{\beta} .
\end{aligned} \tag{20}$$

In the last expression we have used brackets to emphasize the order of operations that leads to an efficient implementation and avoids computing $A = PK_{\cdot+}$. It can be verified that the time required is $O(mm_+)$.

We obtain the Hessian by taking the derivative of (20) with respect to $\boldsymbol{\beta}$:

$$H \triangleq \frac{\partial^2 F}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = \frac{1}{2\epsilon m_+m_-}\big(PK_{\cdot+}I^{\mathcal{Q}}PK_{\cdot+}\big) + \lambda K_{++} .$$

Note the Hessian requires computing A. However, for the linear conjugate gradient method we only require computing $H\mathbf{s}$ for some vector $\mathbf{s}$. In this case, we can avoid computing A by using the following order of operations:

$$H\mathbf{s} = \frac{1}{2\epsilon m_+m_-}\big(P\big(K_{\cdot+}\big(I^{\mathcal{Q}}P\big)(K_{\cdot+}\mathbf{s})\big)\big) + \lambda K_{++}\mathbf{s} .$$

The time required to compute $H\mathbf{s}$ is also $O(mm_+)$.

In the subspace-trust-region method we use a maximum of 25 conjugate gradient iterations.[2] We found the iterations usually converge a solution of acceptable accuracy in a constant bounded number of trust region iterations. Since each iteration requires $O(mm_+)$ time, the total time required by the algorithm is, practically speaking, $O(mm_+)$. Total space is also $O(mm_+)$.

Finally, we note that we can slightly improve the time required to compute the gradient and Hessian-vector product by first sorting the values of $K_{\cdot+}\boldsymbol{\beta}$ or $K_{\cdot+}\mathbf{s}$. Though this does not improve the *big-O* efficiency, it does reduce the constant factor. We refer the interested reader to [12] for details on a method which can be adapted for the nonlinear RankRC objective (13).

6 Experiments

In this section we empirically compare RankRC to other methods on several unbalanced problems. The following methods are compared:

1) KNN: k-Nearest-Neighbors algorithm. The posterior probability is used as the ranking function:

$$P(y|\mathbf{x}) = \frac{1}{k}\sum_{i\in\mathcal{K}} \mathbb{I}[y_i = 1] ,$$

where $\mathcal{K}$ is the set of k nearest neighbors in the training dataset.

2) SVM: This is the standard nonlinear SVM [26], in which the primal problem,

2. We use diagonal preconditioning and warm-starts as λ is varied from high to low.

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m}\max\big(0, 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)\big) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,,$$

is solved (in the dual) to obtain the decision function, $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b = \sum_{i=1}^{m}\beta_i k(\mathbf{x}_i, \mathbf{x}) + b$, with $k(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x})$.

3) SVM-W: Weighted SVM [26], [35] in which

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m}\omega_i\max\big(0, 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)\big) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,,$$

is solved, with different weights associated with each class:

$$\omega_i = \begin{cases} \frac{m}{2m_+} & \text{if } y_i = +1 \\ \frac{m}{2m_-} & \text{if } y_j = -1 \,. \end{cases}$$

The idea is to penalize misclassification error of minority examples more heavily in order to reduce the bias towards majority examples.

4) SVM-RUS: Randomly Under Sample the majority class examples ($y = -1$) to match the number of minority examples [9]. The resulting dataset, with $2m_+$ points, is used to train a standard SVM.

5) SVM-SMT: Uses a Synthetic Minority Oversampling TEchnique (SMOTE) [10] in which the rare class is over-sampled by creating new synthetic rare class samples according to each rare class sample and its k nearest neighbors. Each new sample is generated in the direction of some or all of the nearest neighbors. We oversample to match the number of majority examples. The resulting dataset, with $2m_-$ points, is used to train a standard SVM.

6) RANK-SVM: Nonlinear RankSVM problem (6).

7) RANK-RND: We solve the regularized ranking problem constrained to m_+ randomly selected set of basis function, i.e. Problem (11) with $|\mathcal{R}| = m_+$ and a randomly chosen index set $\mathcal{R}$.

8) RANK-RC: We solve the regularized ranking problem with a Rare Class representation, i.e. Problem (13).

We use LIBSVM [36] to solve the SVM problems (2-5). LIBSVM is a popular and efficient implementation of the sequential minimal optimization algorithm [37]. We set cache size to 10 GB to minimize cache misses; termination criteria and shrinking heuristics are used in their default settings. The ranking methods (6-8) are solved using the subspace-trust-region method as outlined in Section 5. Termination tolerance is set at 1e-6. For ranking methods, the memory available to store the kernel matrix is limited to 10 GB. Experiments are performed on a Xeon E5620@2.4 Ghz running Linux.

All datasets are standardized to zero mean and unit variance before training. Since our focus is on nonlinear kernels, for all SVM and ranking methods (2-8), we use a Gaussian kernel, $k(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|_2^2/\sigma^2)$ with $\sigma^2 = \frac{1}{m^2}\sum_{i,j=1}^{m}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$. The penalty parameter λ is determined by cross-validation over values $\log_2\lambda = [-20, -18, \ldots, 8, 10]$. For KNN we cross-validate over $k = [1, 2, \ldots, \lceil\min(100, \sqrt{m})\rceil]$.

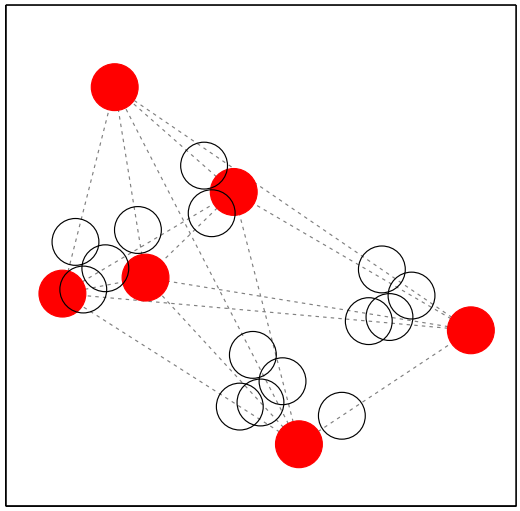


Fig. 4. Example configuration of simulated dataset in two-dimensions with $\sigma = 0.1$ and $t = 0.75$. The red, filled in circles show the locations of the six normal components for the rare class distribution. The black, empty circles show the location of the 15 normal components for the majority class distribution, whose centers lie along the dotted lines connecting all two rare class normal components.

6.1 Simulated Data

Simulated unbalanced datasets are generated in the following manner. Rare class instances are sampled from six multivariate normal distributions with equal probability. Their enters, μ_i, $i = 1, \ldots, 6$, are randomly chosen within a unit cube. The majority class is sampled from $\binom{6}{2} = 15$ multivariate normal distributions with equal probability. Their centers are chosen along lines connecting all combinations of two rare class centers, i.e. $t\mu_i + (1-t)\mu_j$, $i > j$. The parameter $t \in [0, 1]$ is used to roughly control the degree of class overlap. All covariances are chosen to be spherical, $\sigma^2 I$. Finally, the imbalance ratio, $\rho = \frac{m_+}{m}$, is used to determine the number of samples drawn from each of the class conditional distributions. An example configuration in two-dimensional space is shown in Fig. 4. The resulting dataset contains multiple rare-class subconcepts that vary in discriminative structure.

For our experiment we generate data in five-dimensional space with $\sigma = 0.5$. We set $t = 0.9$, 0.75, and 0.6 to produce datasets with high, medium and low overlap, respectively. The imbalance ratio, ρ, is varied from 10 to 40 percent in 10 percent increments for each t value. Thus we have a total of 12 datasets. For each dataset we generate 1,000 training points, 1,000 validation points and 10,000 testing points. Results are averaged over 10 trials.

Table 2 shows test AUC results using different methods. KNN does not perform as well as SVM and ranking methods. In general, ranking methods perform better than SVM based methods when there is greater overlap and higher imbalance (lower ρ). RANK-RND under performs in medium and low overlap datasets. In comparison, RANK-RC yields statistically similar performance as RANK-SVM across all datasets. Overall, both RANK-RC and RANK-SVM provide the best models in terms of AUC.

Fig. 5 further compares the empirical ranking loss function,

TABLE 2
Comparison of Test AUC Results for Simulated Datasets with High ($t = 0.9$), Medium ($t = 0.75$), and Low ($t = 0.6$) Overlap, Each with $\rho = 10\%$, 20%, 30% and 40% Minority Samples

Overlap	ρ	KNN	Classification Loss				Ranking Loss			True Bayes
			SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC	
High	10%	56.3 ± 0.4	57.3 ± 0.3	59.8 ± 0.3	59.1 ± 0.4	58.9 ± 0.4	$\mathbf{61.5 \pm 0.3}$	$\mathbf{61.4 \pm 0.3}$	$\mathbf{61.4 \pm 0.2}$	66.8
	20%	55.5 ± 0.4	59.2 ± 0.2	61.2 ± 0.1	61.0 ± 0.4	60.9 ± 0.4	$\mathbf{61.6 \pm 0.3}$	$\mathbf{61.3 \pm 0.4}$	$\mathbf{62.3 \pm 0.3}$	
	30%	57.3 ± 1.0	61.1 ± 0.2	$\mathbf{62.2 \pm 0.4}$	62.2 ± 0.1	61.8 ± 0.3	$\mathbf{62.3 \pm 0.4}$	$\mathbf{62.2 \pm 0.3}$	$\mathbf{62.6 \pm 0.4}$	
	40%	59.0 ± 0.7	$\mathbf{62.8 \pm 0.2}$	$\mathbf{63.2 \pm 0.3}$	$\mathbf{63.3 \pm 0.2}$	$\mathbf{62.8 \pm 0.2}$	$\mathbf{62.7 \pm 0.3}$	62.5 ± 0.2	$\mathbf{62.7 \pm 0.3}$	
Medium	10%	54.9 ± 0.6	56.8 ± 0.5	59.5 ± 0.5	58.8 ± 0.3	58.6 ± 0.9	$\mathbf{61.4 \pm 0.5}$	60.1 ± 0.5	$\mathbf{61.4 \pm 0.4}$	69.5
	20%	54.5 ± 0.3	60.1 ± 0.4	62.1 ± 0.2	62.1 ± 0.3	62.0 ± 0.5	$\mathbf{62.8 \pm 0.4}$	61.5 ± 0.4	$\mathbf{63.4 \pm 0.2}$	
	30%	57.6 ± 0.8	63.4 ± 0.1	$\mathbf{64.0 \pm 0.3}$	63.3 ± 0.1	63.4 ± 0.4	$\mathbf{64.1 \pm 0.4}$	62.4 ± 0.5	$\mathbf{64.6 \pm 0.5}$	
	40%	58.0 ± 0.7	$\mathbf{65.3 \pm 0.2}$	$\mathbf{65.5 \pm 0.1}$	$\mathbf{65.4 \pm 0.1}$	$\mathbf{65.3 \pm 0.2}$	64.6 ± 0.3	63.0 ± 0.2	$\mathbf{64.9 \pm 0.3}$	
Low	10%	55.9 ± 1.0	61.1 ± 0.6	64.0 ± 0.4	63.5 ± 0.2	63.0 ± 0.8	$\mathbf{65.3 \pm 0.5}$	62.7 ± 0.6	$\mathbf{65.5 \pm 0.4}$	74.4
	20%	57.2 ± 0.4	64.2 ± 0.3	66.6 ± 0.2	66.2 ± 0.2	66.4 ± 0.2	$\mathbf{67.1 \pm 0.3}$	63.5 ± 0.4	$\mathbf{67.3 \pm 0.2}$	
	30%	59.9 ± 0.9	$\mathbf{69.1 \pm 0.4}$	$\mathbf{69.4 \pm 0.2}$	68.9 ± 0.1	$\mathbf{69.2 \pm 0.2}$	$\mathbf{69.2 \pm 0.4}$	65.8 ± 0.4	$\mathbf{69.8 \pm 0.4}$	
	40%	61.6 ± 1.5	$\mathbf{71.1 \pm 0.1}$	$\mathbf{70.7 \pm 0.3}$	70.5 ± 0.1	$\mathbf{71.1 \pm 0.1}$	69.8 ± 0.3	65.8 ± 0.3	69.9 ± 0.3	

Mean AUC score with standard error over 10 trials are shown. Bolded scores indicate the result is statistically not different than the best performing model using a two-tailed t-test with 99 percent confidence.

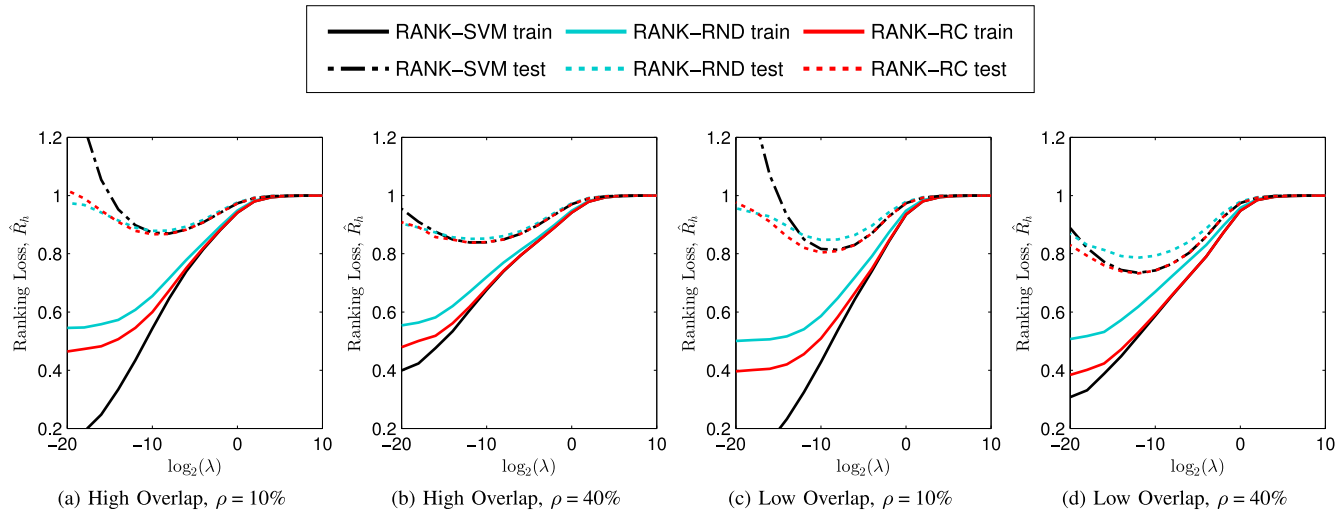
$$\hat{R}_h = \frac{1}{m_+ m_-} \sum_{y_i=+1} \sum_{y_j=-1} \ell_h\big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big) \ ,$$

obtained by the ranking methods on four of the training and testing sets as λ is varied. Firstly we observe that the difference between RANK-SVM and the restricted basis models (RANK-RND and RANK-RC) decreases as λ is increased. Secondly, Fig. 5 shows, more noticeably in Figs. 5c and 5d, that RANK-RND is unable to achieve the optimal test loss levels at moderate values of λ, which are achieved by both RANK-SVM and RANK-RC. Thirdly, the test loss of RANK-RND and RANK-RC is lower than that of RANK-SVM for small values of λ. This suggests that, for RANK-SVM, kernel and penalty parameters can more easily lead to overfitting the rare class samples, even with cross validation. The performance comparison of this example is consistent with the motivations provided in Section 4: RANK-RC models are sufficiently complex for rare class problems and yet they are simpler than the full kernel models, due to the dimension reduction in the feature space. This allows RANK-RC to achieve a similar level of test performance as RANK-SVM but provides additional robustness. This robustness property can be important in practice, given the possibility of overfitting a model through cross-validation.

6.2 Real Datasets

In this section the methods are compared on several unbalanced real datasets obtained from various sources. Table 3 lists the datasets along with their characteristics. For each dataset, three-quarters of the observations are used for training and the remaining one-quarter for out-of-sample testing. Results are averaged over 20 stratified random splits of the data. The model parameter (λ or k) is tuned by running 10-fold cross-validation on the training set for each split.

Table 4 shows the mean test AUC score with standard error for each model. Overall, RANK-SVM and RANK-RC yield the best performing models with statistically similar results. RANK-RND, on the other hand, under performs on some datasets, indicating that a random basis set is not as effective as the rare class basis on unbalanced problems. SVM based methods generally do not perform as well as ranking methods, except when there appears to be more discriminative patterns in the data.



(a) High Overlap, $\rho = 10\%$ (b) High Overlap, $\rho = 40\%$ (c) Low Overlap, $\rho = 10\%$ (d) Low Overlap, $\rho = 40\%$

Fig. 5. Comparison of empirical train and test ranking loss obtained by the ranking methods on four of the simulated datasets as λ is varied.

TABLE 3
List of Datasets and Their Characteristics that We Use to Evaluate Methods

Name	Source	Features		Samples		
		Original	d	m	m_+	ρ
Abalone19	UCI	1N,7C	10	4,177	32	0.8%
Mammograph	[38]	6C	6	11,183	260	2.3%
Ozone	UCI	72C	72	2,536	73	2.9%
YeastME2	UCI	8C	8	1,484	51	3.4%
Wine4	UCI	11C	11	4,898	183	3.7%
Oil	[39]	49C	49	937	41	4.4%
SolarM0	UCI	10N	32	1,389	68	4.9%
Coil	KDD	85C	85	9,822	586	6.0%
Thyroid	UCI	21N,7C	52	3,772	231	6.1%
Libras	UCI	90C	90	360	24	6.7%
Scene	LibSVM	294C	294	2,407	177	7.4%
YeastML8	LibSVM	103C	103	2,417	178	7.4%
Crime	UCI	122C	100	1,994	150	7.5%
Vowel0	Keel	10C	10	989	90	9.1%
Euthyroid	UCI	18N,7C	42	3,163	293	9.3%
Abalone7	UCI	1N,7C	10	4,177	391	9.4%
Satellite	UCI	36C	36	6,435	626	9.7%
Page0	Keel	10C	10	5,472	559	10.2%
Ecoli	UCI	7C	7	336	35	10.4%
Contra2	Keel	9C	9	1,473	333	22.6%

Under original features, 'N' is used to denote number of nominal features, 'C', is used to denote number of continuous features. We derive d features by converting nominal features to an indicator representation and use continuous features as is. Under samples, m is the total number of observations, m_+ is the number of rare class observations, and $\rho = \frac{m_+}{m}$ is the percentage of rare class examples.

Table 5 compares the number of support vectors used by the SVM and ranking models. RANK-SVM uses more support vectors than SVM based models. It can use even more support vectors than SVM-SMT, which is trained on an enlarged dataset almost twice the size. This suggests that training RANK-SVM using a working-set type algorithm, which only tracks active support vectors (e.g. as proposed in [27] for standard SVMs), would still run costly in time and space. In comparison, RANK-RND and RANK-RC use significantly fewer support vectors, an indication of better robustness. Moreover, with RANK-RC, test performance is also not compromised.

6.3 Intrusion Detection

In this section we use the KDD Cup 1999 dataset [40] to evaluate a large-scale unbalanced problem. The objective is to detect network intrusion by distinguishing between legitimate (normal) and illegitimate (attack) connections to a computer network. The dataset is a collection of simulated raw TCP dump data over a period of nine weeks on a local area network. The first seven weeks of data is used for training and the last two for test, providing a total of 4,898,431 training observations and 3,11,029 test cases. We processed the data to remove duplicate entries (as done in [41]) resulting in 1,074,975 training observations and 77,286 test cases. Each observation contains 41 features, three of which are categorical and the rest numerical. The three categorical features are protocol (three categories), service (70 categories) and flag (11 categories). We represent protocol using three binary features, where each feature is an indicator for one of the three categories. Service and flag categories are replaced by the frequency in the training sample (i.e. probability) corresponding to the event of observing an attack given the category is present. Thus we obtain a total of 43 features. Finally, as done for all datasets, we standardize each feature to zero mean and unit variance.

The attack types are grouped in four categories, DOS (Denial of Service), Probing (Surveillance, e.g. port scanning), U2R (user to root), R2L (remote to local). Table 6 shows the distribution of attack types in the training and test sets. Together, the U2R and R2L attacks constitute

TABLE 4
Comparison of Test AUC Results for Real Datasets (Listed in Table 3)

Dataset	KNN	Classification Loss				Ranking Loss		
		SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC
Abalone19	55.7 ± 2.2	54.9 ± 3.1	64.3 ± 1.3	74.1 ± 1.5	67.4 ± 1.1	$\mathbf{81.0 \pm 1.2}$	$\mathbf{79.1 \pm 1.1}$	$\mathbf{81.4 \pm 1.1}$
Mammograph	80.7 ± 0.4	88.4 ± 0.4	90.1 ± 0.7	92.8 ± 0.4	91.3 ± 0.4	$\mathbf{93.7 \pm 0.4}$	$\mathbf{93.9 \pm 0.4}$	$\mathbf{94.4 \pm 0.3}$
Ozone	66.4 ± 2.2	85.0 ± 1.1	85.5 ± 0.7	86.4 ± 0.9	85.9 ± 0.8	$\mathbf{89.4 \pm 0.9}$	$\mathbf{88.7 \pm 0.8}$	$\mathbf{90.1 \pm 0.9}$
YeastME2	69.3 ± 1.7	81.8 ± 0.5	85.5 ± 0.7	$\mathbf{87.5 \pm 0.7}$	$\mathbf{86.8 \pm 1.1}$	$\mathbf{90.8 \pm 0.8}$	$\mathbf{89.0 \pm 0.9}$	$\mathbf{89.4 \pm 1.1}$
Wine4	61.9 ± 0.5	74.9 ± 0.7	71.6 ± 0.9	79.1 ± 0.8	78.9 ± 0.8	$\mathbf{83.5 \pm 0.6}$	79.6 ± 0.6	$\mathbf{82.7 \pm 0.7}$
Oil	71.6 ± 1.7	$\mathbf{91.1 \pm 0.9}$	$\mathbf{88.0 \pm 1.4}$	$\mathbf{90.6 \pm 0.8}$	$\mathbf{90.6 \pm 1.0}$	$\mathbf{92.5 \pm 0.9}$	89.2 ± 1.0	$\mathbf{91.7 \pm 0.8}$
SolarM0	58.9 ± 1.6	55.4 ± 0.7	63.1 ± 1.3	71.5 ± 0.8	73.1 ± 0.4	$\mathbf{78.5 \pm 0.5}$	$\mathbf{77.2 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$
Coil	53.9 ± 0.3	59.2 ± 0.8	62.9 ± 0.4	68.8 ± 0.4	67.5 ± 0.5	70.0 ± 0.4	69.8 ± 0.4	$\mathbf{72.3 \pm 0.2}$
Thyroid	73.9 ± 0.6	$\mathbf{94.8 \pm 0.4}$	93.4 ± 0.5	$\mathbf{94.8 \pm 0.3}$	$\mathbf{94.4 \pm 0.4}$	$\mathbf{95.7 \pm 0.4}$	91.3 ± 0.5	$\mathbf{95.7 \pm 0.3}$
Libras	87.4 ± 2.1	$\mathbf{96.8 \pm 0.9}$	$\mathbf{96.7 \pm 0.9}$	$\mathbf{96.4 \pm 0.8}$	$\mathbf{96.8 \pm 0.9}$	$\mathbf{97.6 \pm 0.8}$	$\mathbf{95.4 \pm 1.0}$	$\mathbf{94.8 \pm 1.1}$
Scene	59.3 ± 0.8	67.3 ± 0.8	$\mathbf{75.4 \pm 0.9}$	74.8 ± 0.6	74.0 ± 0.9	$\mathbf{77.1 \pm 0.7}$	$\mathbf{76.4 \pm 0.8}$	$\mathbf{77.5 \pm 0.6}$
YeastML8	54.5 ± 0.7	57.1 ± 0.8	59.6 ± 0.6	57.9 ± 0.5	59.7 ± 0.4	$\mathbf{61.5 \pm 0.5}$	$\mathbf{60.2 \pm 0.7}$	$\mathbf{62.0 \pm 0.5}$
Crime	71.8 ± 1.5	87.6 ± 0.7	87.3 ± 0.6	90.1 ± 0.3	90.8 ± 0.3	$\mathbf{92.3 \pm 0.3}$	$\mathbf{91.2 \pm 0.3}$	$\mathbf{91.6 \pm 0.3}$
Vowel0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.8 ± 0.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	98.4 ± 0.1	$\mathbf{100.0 \pm 0.0}$
Euthyroid	75.8 ± 0.8	$\mathbf{95.0 \pm 0.4}$	$\mathbf{95.0 \pm 0.4}$	$\mathbf{94.6 \pm 0.4}$	$\mathbf{95.0 \pm 0.4}$	$\mathbf{95.2 \pm 0.4}$	90.7 ± 0.4	$\mathbf{94.1 \pm 0.4}$
Abalone7	78.2 ± 2.1	56.1 ± 3.2	76.3 ± 0.5	77.4 ± 0.3	74.4 ± 0.2	$\mathbf{87.0 \pm 0.3}$	$\mathbf{86.5 \pm 0.3}$	$\mathbf{87.1 \pm 0.3}$
Satellite	83.8 ± 0.3	$\mathbf{94.8 \pm 0.1}$	94.6 ± 0.1	94.3 ± 0.1	$\mathbf{95.1 \pm 0.1}$	$\mathbf{95.3 \pm 0.1}$	94.3 ± 0.1	$\mathbf{95.1 \pm 0.1}$
Page0	90.4 ± 0.4	$\mathbf{98.4 \pm 0.1}$	98.1 ± 0.1	98.1 ± 0.1	98.2 ± 0.1	$\mathbf{98.6 \pm 0.1}$	95.6 ± 0.1	$\mathbf{98.4 \pm 0.1}$
Ecoli	75.6 ± 2.0	$\mathbf{94.6 \pm 0.7}$	$\mathbf{93.7 \pm 0.7}$	$\mathbf{94.1 \pm 0.6}$	$\mathbf{93.2 \pm 0.6}$	$\mathbf{94.1 \pm 0.6}$	$\mathbf{93.4 \pm 0.9}$	$\mathbf{94.5 \pm 0.7}$
Contra2	60.5 ± 0.9	66.9 ± 0.8	70.2 ± 0.5	70.5 ± 0.6	70.6 ± 0.8	$\mathbf{73.2 \pm 0.5}$	$\mathbf{72.6 \pm 0.5}$	$\mathbf{73.4 \pm 0.4}$

Mean AUC score with standard error over 20 trials are shown. Each trial uses one-quarter data for out-of-sample testing. Bolded scores indicate the result is statistically not different than the best performing model using a two-tailed t-test with 99 percent confidence.

TABLE 5
Average Number of Support Vectors Used by the SVM and Ranking Models over 20 Trials

Dataset	Classification Loss				Ranking Loss		
	SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC
Abalone19	117	1,555	32	2,644	2,979	24	24
Mammograph	306	2,152	119	2,987	7,252	195	193
Ozone	206	746	61	1,165	995	55	55
YeastME2	105	429	41	594	1,066	38	38
Wine4	458	2,109	179	3,166	3,550	136	136
Oil	84	311	32	340	488	31	31
SolarM0	183	845	90	1,114	1,042	51	51
Coil	1,754	5,560	704	8,178	7,284	435	435
Thyroid	332	723	137	1,020	2,739	168	172
Libras	35	93	22	124	197	18	18
Scene	603	1,171	213	1,566	1,748	132	133
YeastML8	833	1,669	257	1,562	1,804	133	133
Crime	308	631	115	867	1,326	112	112
Vowel0	35	37	25	45	730	68	67
Euthyroid	389	673	177	1,002	2,303	216	219
Abalone7	713	1,391	274	2,076	3,079	291	292
Satellite	773	1,158	301	1,526	4,734	466	469
Page0	322	570	145	924	4,012	415	416
Ecoli	47	68	18	115	248	26	26
Contra2	560	843	396	912	1,096	249	249

For ranking models, support vectors are counted as the number of non-zero coefficients associated with kernel functions.

4.0 percent of the test dataset, which is a substantial increase compared to the training set, but still a small fraction. Poor results have been reported in literature for identifying the U2R and R2L attacks [42]. In this experiment, we focus on identifying these attack types by forming a binary classification problem with the positive class representing either a U2R or R2L attack, and the negative class representing all other connection types. Thus the final training set is highly skewed with only 0.098 percent positive instances.

We train using 5, 10, 25, 50, and 75 percent of the training data. The remaining training data is used for validation. We are unable to train RANK-SVM, even with just 5 percent of the data (53,749 samples), since the kernel matrix is too large to store in memory (>10 GB). Clearly, this is an example where a large-scale solution is necessary to solve the ranking problem. We do not train SVM-SMT due to the large number of samples as well. We are able to train SVM-W using up to 50 percent of the data. With more samples SVM-W does not converge, due to the large number of support vectors which do not fit in the cache.

Fig. 6a shows test AUC results obtained by different methods as training data is increased. We observe that SVM and SVM-RUS perform poorly. RANK-RC, RANK-RND and SVM-W produce better results, with RANK-RC performing the best.

TABLE 6
Distribution of Connection Types in Training and Test Sets for the Intrusion Detection Problem

	Training		Test	
Normal	812,808	75.6%	47,913	62.0%
DOS	247,266	23.0%	23,568	30.5%
Probing	13,850	1.3%	2,677	3.5%
U2R	52	0.005%	215	0.278%
R2L	999	0.093%	2,913	3.769%
Total	1,074,975	100%	77,286	100%

Figs. 6b and 6c compare training time and number of support vectors, respectively, as training data is increased. SVM and SVM-RUS train in reasonable time, though they do not produce good models. On the other hand, SVM-W quickly becomes very expensive. RANK-RC and RANK-RND scale well, while able to produce effective models. RANK-RC and RANK-RND also use significantly fewer support vectors than SVM-W.

7 Conclusion

In this paper, we use a ranking loss function to tackle the problem of learning from unbalanced datasets. Minimizing biclass ranking loss is equivalent to maximizing the AUC measure, which overcomes the inadequacies of accuracy, used by conventional classification algorithms. The resulting regularized loss minimization problem corresponds to a biclass RankSVM problem. We modify RankSVM to take advantage of the rare class situation by restricting the solution to a linear combination of rare class kernel functions (RankRC). Assuming that both the number conjugate gradient iterations required for each trust region subproblem and the number of trust region subproblems required for solving a regularized problem are bounded by constants (which are often the case in practice), it allows us to solve the nonlinear ranking problem in $O(mm_+)$ time and $O(mm_+)$ space in practice, thus enabling us to solve many problems which are too large for kernel RankSVM.

We illustrate that, for rare class problems, the proposed rare class representation is sufficiently complex but has better robustness properties, since it corresponds to a dimension reduction in the feature space based on the rare class samples. Using both synthetic and real data sets, we demonstrate computationally that, for rare class problems, RankSVM and RankRC yield better performance in AUC, in comparison to SVM methods, KNN, and RANK-RND. Comparing to RankSVM, while performing similarly in the AUC

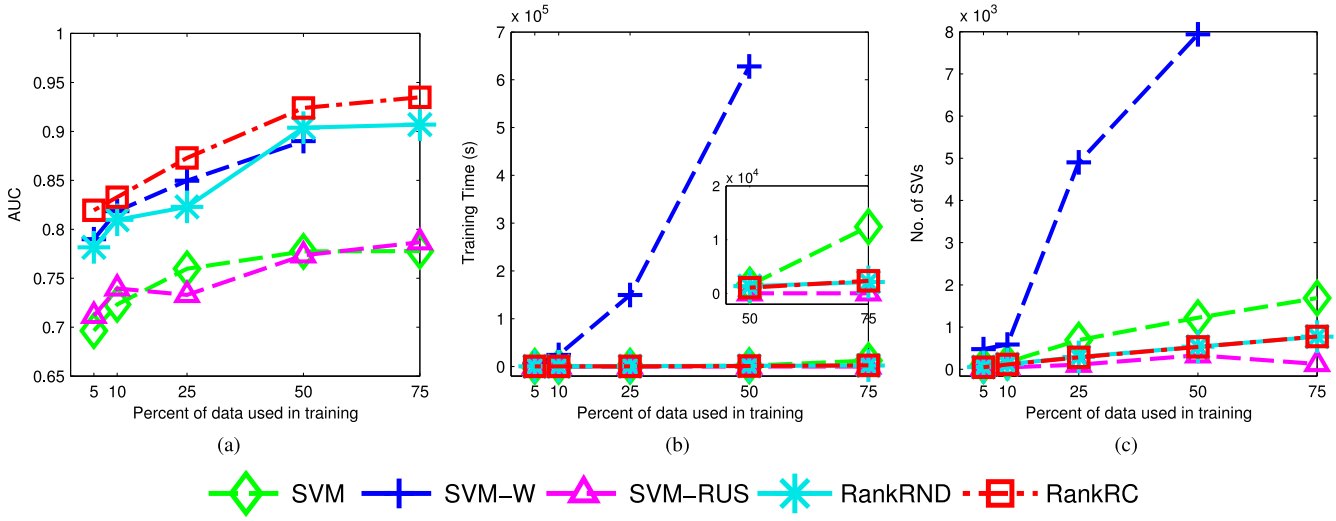


Fig. 6. Comparison of (a) test AUC score, (b) training time in seconds, and (c) number of support vectors, for the intrusion detection problem as percent of data used for training is increased from 5 to 75 percent. In our experiment setup, we were unable to train RANK-SVM due to the large size of the dataset. Also, for more than 50 percent of data, SVM-W did not converge after more than 72 hours of training.

measure, RankRC is shown to require less number of support vectors and be more robust with respect to parameter cross validations. In addition, RankRC is computationally significantly more efficient with respect to both time and space requirements. In particular RankRC yields excellent solutions to the network intrusion problem, while RankSVM is unable to train even with 5 percent of the data set.

Finally we list a few extensions/variants one may consider using the rare class representation:

1) *Regularization*: In problem (13) we can use an ℓ_1-regularizer, $\|\boldsymbol{\beta}\|_1$, instead of $\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta}$. This would lead to sparser solutions [43] and could be solved using coordinate descent methods [44].
2) *Loss function*: We can replace the loss function with other variants of ranking loss. The AUC concentrates uniformly across all threshold levels. We can use weighted AUC [45] or the p-norm push [46] to emphasize specific portions of the AUC curve. Also, we can use list based ranking methods to optimize other criteria such as F_1-score or Precision/Recall breakeven point [47]. The rare-class representation allows us to learn a nonlinear function for unbalanced datasets with more complex loss functions, in reasonable time and space.
3) *Stochastic learning*: For very large datasets, the $m \times m_+$ kernel submatrix may be too large to fit in memory. In this case, we can store $K_{++} \in \mathbb{R}^{m_+ \times m_+}$ and cycle (randomly) through majority class examples updating the $\boldsymbol{\beta} \in \mathbb{R}^{m_+}$ vector via gradient descent using an adaptive learning rate [48]. Unlike standard stochastic gradient descent, in each iteration we use the full set of minority examples and a single (or small subset) of majority samples to perform the update. This should lead to faster convergence while using only $O(m_+ m_+)$ space.
4) *Feature selection*: Selecting appropriate input features can improve model quality significantly, particularly when rare class concepts are embedded in majority examples. For this purpose, we can extend the primal kernel feature selection method proposed by [49] for RankRC. Since we restrict the hypothesis to use rare class kernel functions, overfitting due to feature selection is also less likely.

In summary, the rare class representation offers significant benefits to learn nonlinear models for large-scale rare class problems.

ACKNOWLEDGMENTS

The authors acknowledge comments from anonymous referees which have significantly improved presentation of the paper. All of the authors acknowledge funding from the National Sciences and Engineering Research Council of Canada. Thomas F. Coleman acknowledges funding from the Ophelia Lazaridis University Research Chair. The views expressed herein are solely from the authors.

REFERENCES

[1] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, 2002.
[2] B. Raskutti and A. Kowalczyk, "Extreme re-balancing for SVMS: A case study," *SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 60–69, 2004.
[3] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 49–56.
[4] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, 2004.
[5] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 1–6, 2004.
[6] G. M. Weiss, "Mining with rarity: A unifying framework," *SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 7–19, 2004.
[7] K. Ezawa, M. Singh, and S. W. Norton, "Learning goal oriented Bayesian networks for telecommunications risk management," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 139–147.
[8] J. Zhang and I. Mani, "KNN approach to unbalanced data distributions: A case study involving information extraction," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 42–48.
[9] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. Int. Conf. Mach. Learn.*, 1997, pp. 179–186.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.
[11] R. Herbrich, T. Graepel, and K. Obermayer, *Large Margin Rank Boundaries for Ordinal Regression*. Cambridge, MA, USA: MIT Press, 2000.
[12] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with SVMs," *Inf. Retr.*, vol. 13, no. 3, pp. 201–215, 2010.
[13] M. Zhu, W. Su, and H. A. Chipman, "LAGO: A computationally efficient approach for statistical detection," *Technometrics*, vol. 48, pp. 193–205, 2006.
[14] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proc. 15th Int. Conf. Mach. Learn.*, 1997, pp. 445–453.
[15] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 73–80.
[16] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recogn.*, vol. 40, no. 12, pp. 3358–3378, 2007.
[17] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[18] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recog.*, vol. 30, pp. 1145–1159, 1997.
[19] C. E. Metz, "Basic principles of ROC analysis," *Seminars Nuclear Med.*, vol. 8, no. 4, pp. 283–298, 1978.
[20] J. A. Hanley and B. J. Mcneil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[21] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson, "Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach," *Biometrics*, vol. 44, no. 3, pp. 837–845, 1988.
[22] J. M. Lobo, A. Jiménez-Valverde, and R. Real, "AUC: A lisleading measure of the performance of predictive distribution models," *Ecological Sounding*, vol. 17, pp. 145–7151, 2008.
[23] D. Hand, "Measuring classifier performance: A coherent alternative to the area under the ROC curve," *Mach. Learn.*, vol. 17, pp. 103–123, 2009.
[24] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *J. Am. Statistical Assoc.*, vol. 101, no. 473, pp. 138–156, 2006.
[25] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 15th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.
[26] V. N. Vapnik, *Statistical Learning Theory*, 1st ed. New York, NY, USA: Wiley, 1998.
[27] O. Chapelle, "Training a support vector machine in the primal," *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, 2007.
[28] G. Kimeldorf and G. Wahba, "A correspondence between Bayesian estimation of stochastic processes and smoothing by splines," *Ann. Math. Statist.*, vol. 41, pp. 495–502, 1970.
[29] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. 14th Annu. Conf. Comput. Learn. Theory 5th Eur. Conf. Comput. Learn. Theory*, 2001, pp. 416–426.
[30] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 133–142.
[31] A. Rakotomamonjy, "Optimizing area under ROC curve with SVMs," in *Proc. Workshop ROC Anal. Artif. Intell.*, 2004, pp. 71–80.
[32] K. Ataman, W. N. Street, and Y. Zhang, "Learning to rank by maximizing auc with linear programming," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2006, pp. 123–129.
[33] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[34] M. A. Branch, T. F. Coleman, and Y. Li, "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems," *SIAM J. Sci. Comput.*, vol. 21, no. 1, pp. 1–23, 1999.
[35] E. E. Osuna, R. Freund, and F. Girosi, "Support vector machines: Training and applications," MIT, Cambridge, MA, USA, Tech. Rep. AIM-1602, 1997.
[36] C.-C. Chang and C.-J. Lin. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* [Online]. vol. 2, pp. 27:1–27:27. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm
[37] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.
[38] K. Woods, C. Doss, K. Bowyer, J. Solka, C. Preibe, and P. Keglmyer, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *Int. J. Pattern Recog. Artif. Intell.*, vol. 7, pp. 1417–1436, 1993.
[39] M. Kubat, R. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Mach. Learn.*, vol. 30, pp. 195–215, 1998.
[40] KDD Cup 1999 [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2015.
[41] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD cup 99 data set," in *Proc. 2nd IEEE Int. Conf. Comput. Intell. Security Defense Appl.*, 2009, pp. 53–58.
[42] M. Sabhnani, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 209–215.
[43] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.
[44] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. Statistical Softw.*, vol. 33, pp. 1–22, 20010.
[45] C. G. Weng and J. Poon, "A new evaluation measure for imbalanced datasets," in *Proc. 7th Australasian Data Mining Conf.*, 2008, vol. 87, pp. 27–32.
[46] C. Rudin, "The P-norm push: A simple convex ranking algorithm that concentrates at the top of the list," *J. Mach. Learn. Res.*, vol. 10, pp. 2233–2271, 2009.
[47] T. Joachims, "A support vector method for multivariate performance measures," in *Proc. 2nd Int. Conf. Mach. Learn.*, 2005, pp. 377–384.
[48] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 161–168.
[49] A. Tayal, T. F. Coleman, and Y. Li, "Primal explicit max margin feature selection for nonlinear support vector machines," *Pattern Recog.*, vol. 47, pp. 2153–2164, 2014.

Aditya Tayal received the BASc degree in computer engineering and the MMath degree in statistics-finance from the University of Waterloo, Waterloo, ON, Canada, in 2003 and 2009, respectively. He is currently working toward the PhD degree in scientific computing, Cheriton School of Computer Science, University of Waterloo. His research interests include solving nonconvex optimization problems in machine learning, large-scale algorithms, and applications in finance and bioinformatics.

Thomas F. Coleman holds the Ophelia Lazaridis University research chair at the University of Waterloo. From 1981 to 2005, he was a professor in the Computer Science Department, Cornell University. He has been a professor at Combinatorics & Optimization, University of Waterloo, since 2005. He is the author of two books on computational mathematics, the editor of six proceedings, and has published more than 70 journal articles in the areas of optimization, applications, automatic differentiation, parallel computing, and computational finance.

Yuying Li has been a professor in the Cheriton School of Computer Science, University of Waterloo, Canada, since 2005. From 1988 to 2005, she was a research associate in the Computer Science Department, Cornell University. She received the 1993 Leslie Fox first Prize in numerical analysis at Oxford England. She has published many papers in optimization, computational finance, and data mining. She has been an associate editor for the *Journal of Computational Finance* since January 2008.