RankRC: Large-scale Nonlinear Rare Class Ranking

Aditya Tayal[a,1,*], Thomas F. Coleman[b,1,2], Yuying Li[a,1]

[a]*Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada N2L 3G1*
[b]*Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada N2L 3G1*

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 an explosion of data in recent years, resulting in many large scale rare class problems. In this paper, we consider nonlinear kernel based classification methods expressed as a regularized loss minimization problem. We address 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 our algorithm RankRC. We provide heuristic and theoretical justification for the rare class representation, and experimentally illustrate the effectiveness of RankRC in both test performance and computational complexity on several datasets.

Keywords: rare class, large scale, nonlinear kernel, receiver operator characteristic, ranking svm

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 [4, 5, 1, 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,

*Corresponding author

Email addresses: amtayal@uwaterloo.ca (Aditya Tayal), tfcoleman@uwaterloo.ca (Thomas F. Coleman), yuying@uwaterloo.ca (Yuying Li)

[1]All three authors acknowledge funding from the National Sciences and Engineering Research Council of Canada

[2]This author acknowledges funding from the Ophelia Lazaridis University Research Chair. The views expressed herein are solely from the authors.

they have little impact on accuracy or total error. Thus majority examples overshadow the minority class, resulting in models which 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 been showed to achieve 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 this paper we focus on nonlinear kernel based classification methods expressed as a regularized loss minimization problem. In recent years, we have 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. 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.

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].
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, Chapelle and Keerthi [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, 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. We present heuristic and theoretical justification for this choice. Specifically, we show RankRC is optimal with respect to RankSVM for skewed data when a subset of kernel functions are used. We also draw connections to the Nyström approximation method. Several of our results are general and can be applied to other regularized loss minimization problems.

The rest of the paper is organized as follows. Sections 2 and 3 review the AUC measure and RankSVM. Section 4 develops the RankRC problem. 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 modeling. For classification problems, error rate is the most commonly used metric. For simplicity, we will consider the two-class case. Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$ 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,

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

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 [13, 14, 15, 16]. 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 majority examples and m_- the number of minority 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.

		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_-

Table 1: Add caption

The Receiver Operating Characteristic (ROC) can be used to obtain a skew independent measure [13, 17, 18]. Most 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 Figure 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.

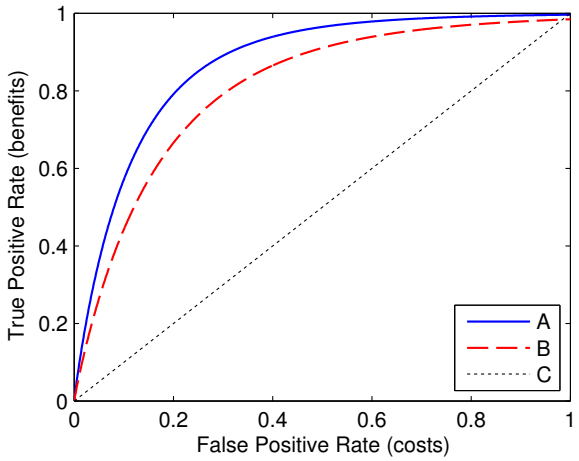


Figure 1: Example ROC curves. Curve A dominates B and curve B dominates C. Curve C has an AUC of 0.5 and indicates a model with no discriminative value.

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 [19]. 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 [20]:

$$AUC = \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \mathbb{I}\left(f(\mathbf{x}_i) > f(\mathbf{x}_j)\right) . \tag{2}$$

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 [21]. Specifically, using the hinge loss, $\ell_h(z) = \max(0, 1-z)$, with ℓ_2-regularization leads to the well known support vector machine (SVM) formulation [22, 23],

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

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:

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

Problem (4) is a special case of RankSVM proposed by Herbrich et al. [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 [22], 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 the optimization procedure has cubic complexity in the number of variables, the complexity of the dual method becomes $O(m^6)$, which is unreasonable for even medium sized datasets.

As noted by Chapelle [24], Chapelle and Keerthi [12], we can also solve the primal problem in the implicit feature space due to the Representer Theorem [25, 26]. 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:

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

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

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

or more simply,

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

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

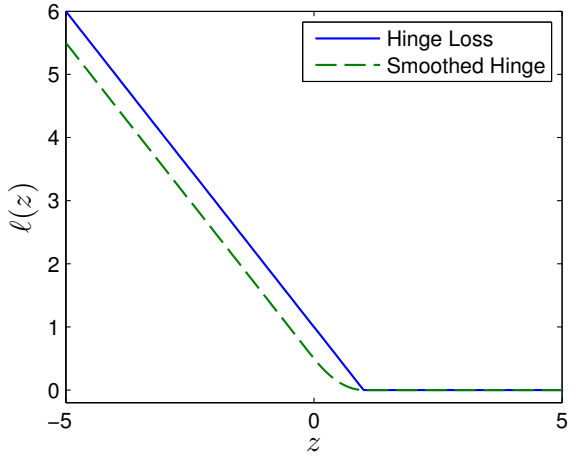


Figure 2: The smoothed hinge is a differentiable approximation of the hinge loss. Here the smoothed hinge is shown with $\epsilon = 0.5$.

approximation, ℓ_ϵ, defined as,

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

which transitions from linear cost to zero cost using a quadratic segment (see Figure 2) and provides similar benefits as the hinge loss. Now 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 [e.g. see 27]. 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. But solving a nonlinear RankSVM in $O(m^3)$ time may not be practical for mid- to large-sized datasets. Moreover, the method requires $O(m^2)$ space to store the kernel matrix. 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 the unbalanced nature of the problem to achieve $O(mm_+)$ time and $O(mm_+)$ space, while not sacrificing performance.

4. RankRC: Ranking with Rare Class Representation

For highly unbalanced datasets, to make SVM computational feasible for large scale problems, we propose a rare class (RC) based method. Specifically we propose a RC based method, which restricts the solution to the form

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

so it consists only of kernel function realizations of the minority class.

Next we present motivate form (7) by assuming specific properties of the class conditional distributions and kernel function. Zhu et al. [28] make use of similar assumptions, however, in their method they attempt to directly estimate the likelihood ratio. In contrast, we use a regularized loss minimization approach.

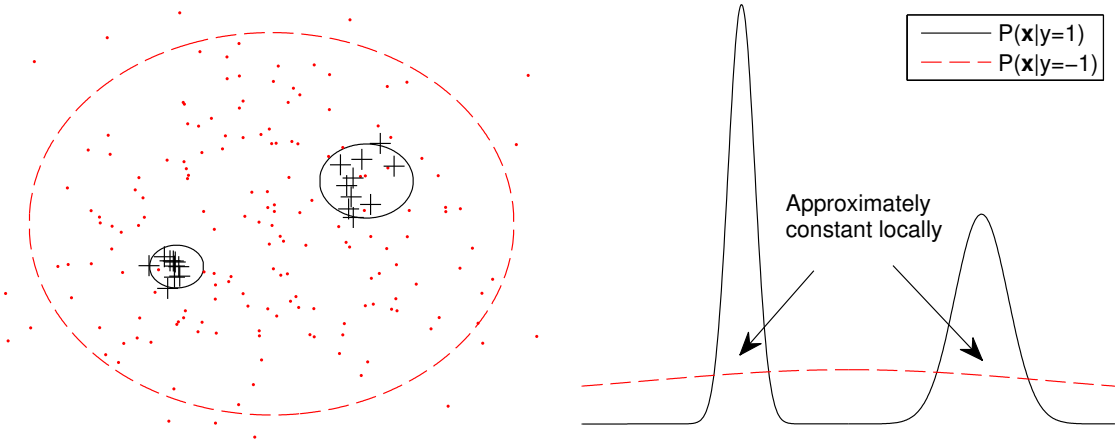
Recall that the optimal ranking function for a classification problem is the posterior probability, $P(y = 1|\mathbf{x})$, since it minimizes the Bayes risk for arbitrary costs. From Bayes' Theorem, we have

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

Any monotonic transformation of (8) also yields equivalent ranking capability. Dividing the numerator and denominator of (8) by $P(y = -1)P(\mathbf{x}|y = -1)$, we note that $P(y = 1|\mathbf{x})$ is a monotonic transformation of the likelihood ratio, denoted as

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

which is the ranking function we wish to obtain. Now, if we assume that the conditional density, $P(\mathbf{x}|y = 1)$, is a mixture of m_+ identical spherical normals centered at the rare class examples, we can write



(a) (b)

Figure 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.

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

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

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

Observe that in rare class problems most examples are from the majority class ($y=-1$) and only a small number are from the rare class ($y=+1$). Therefore it is reasonable to assume $P(\mathbf{x}|y=-1)$ is locally constant in a neighborhood around the minority class examples, see Figure 3 for an illustration. Let $P(\mathbf{x}|y=-1) \approx c_i$ for each minority example i in the neighbourhood of $\mathbf{x}_i$.[3] Then (9) can be written as,

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

[3]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. If the assumptions made are relaxed, we may still expect the rare class representation to perform reasonably well.

For any general regularized loss minimization problem with any loss function $L : \mathbb{R}^m \to \mathbb{R}$, we can consider a corresponding rare class regularization problem. Assume that a penalty parameter $\lambda \in \mathbb{R}_+$ is given, a regularized loss minimization problem can be described as

$$\min_{\mathbf{w}\in\mathbb{R}^{d'}} \quad L(f(\mathbf{x}_1),...,f(\mathbf{x}_m)) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,, \tag{10}$$

where $f : \mathcal{X} \to \mathbb{R}$ is a linear hypothesis in feature space, $f(\mathbf{x})) = \mathbf{w}^T\phi(\mathbf{x})$. Here $L(\cdot)$ is any loss function including both standard SVM and ranking SVM functions, since SVM-Rank is equivalent to a 1-class SVM on an enlarged dataset with the set of points $\mathcal{P} = \{\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) : y_i > y_j, i, j = 1...,m\}$. From the Representer Therorem, a solution vector $\mathbf{w} \in \mathcal{S} = \text{span}\{\phi(\mathbf{x}_i) : i = 1,...,m\}$ can be expressed in terms of all the given training points in the feature space.

Using the restricted hypothesis (7), we consider the following constrained regularized ranking problem,

$$\begin{aligned} &\min_{\beta\in\mathbb{R}^{\mathcal{R}}} \quad L(f(\mathbf{x}_1),...,f(\mathbf{x}_m)) + \frac{\lambda}{2}\beta^T K_{\mathcal{R}\mathcal{R}}\beta \,, \\ &\text{subject to} \;\; f(\mathbf{x}) = \sum_{i\in\mathcal{R}} \beta_i k(\mathbf{x}_i, \mathbf{x}) \end{aligned} \tag{11}$$

where $\mathcal{R} \subseteq \{1,...,m\}$. The proposed ranking with a rare class representation, subsequently referred to as RankRC, is a special case of (11) with $\mathcal{R} = \{i : y_i = 1\}$.

In order to see potential advantages of RankRC, we compare the full data set regularized SVM ranking problem

$$\begin{aligned} &\min_{\beta\in\mathbb{R}^{m}} \quad \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h\left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right) + \frac{\lambda}{2}\beta^T K\beta \\ &\text{subject to} \;\; f(\mathbf{x}) = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \mathbf{x}) \end{aligned} \tag{12}$$

with the subset data regularization problem (11).

Applying Theorem 2 in [**?**], we can establish the following theoretical result.

Theorem 1. *Let $f^*(\mathbf{x})$ be the optimal hypothesis of the full data set SVM-Rank problem* (12) *under the feature mapping $\phi : \mathcal{X} \to \mathcal{F}$ and $\bar{f}^*(\mathbf{x})$ be the optimal hypothesis for the subset data set SVM-Rank problem* (11) *where $\mathcal{R} \subseteq \{1,...,m\}$. Assume there exists $\kappa > 0$ such that $k(\mathbf{x},\mathbf{x}) \leq \kappa$, where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the kernel map associated with ϕ. Then the following inequality holds for all $\mathbf{x} \in \mathcal{X}$:*

$$|f^*(\mathbf{x}) - \bar{f}^*(\mathbf{x})| \leq \frac{2\kappa}{\lambda} \left(\sum_{\{i:y_i=+1\}} \frac{\mathbb{I}[i \notin \mathcal{R}]}{m_+} + \sum_{\{j:y_j=-1\}} \frac{\mathbb{I}[j \notin \mathcal{R}]}{m_-} \right)^{\frac{1}{2}}, \tag{13}$$

where $\mathbb{I}[p]$ denotes the indicator function and is equal to 1 if p is true, 0 if p is false.

We note, the difference in hypothesis decreases for larger regularization according to $O\left(\frac{1}{\lambda}\right)$ and as we include more kernel function realizations in our representation. However, for the ranking loss, the bound decreases asymmetrically depending on whether we include a point from the positive or negative class. In particular, if the dataset is unbalanced with $m_- \gg m_+$, then $\frac{1}{m_+} \gg \frac{1}{m_-}$, and the reduction obtained from including a positive class basis is much greater than including one from the negative class. Hence, for a fixed number of kernel function realizations, the bound is minimized by first including bases corresponding to the positive or rare class.

5. Computational Complexity Comparison between RankRC over SVM-Rank

Using a smooth loss function in (**??**), we have the following RankRC problem in m_+ variables,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{m_+}} \quad \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_\epsilon \left(\boldsymbol{\beta}^T K_{i+} - \boldsymbol{\beta}^T K_{j+} \right) + \frac{\lambda}{2} \boldsymbol{\beta}^T K_{++} \boldsymbol{\beta} \,. \tag{14}$$

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. We also replace ℓ_h with the smooth approximation ℓ_ϵ. To solve (14) 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

$$\boldsymbol{\beta} = U \Lambda^{\dagger \frac{1}{2}} \mathbf{w} \,, \tag{15}$$

where $\Lambda^\dagger$ denotes the pseudoinverse of Λ. We can substitute (15) in (14) to obtain the following problem,

$$\min_{\mathbf{w} \in \mathbb{R}^{m_+}} \quad \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_\epsilon \left(\mathbf{w}^T \Lambda^{\dagger \frac{1}{2}} U^T K_{i+} - \mathbf{w}^T \Lambda^{\dagger \frac{1}{2}} U^T K_{j+} \right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \,, \tag{16}$$

which is a problem in linear space. That is, Problem (16) is equivalent to Problem (4) with data points given by $\mathbf{x}_i = \Lambda^{\dagger\frac{1}{2}} U^T K_{i+} \in \mathbb{R}^{m_+}$, $i = 1, ..., m$. Therefore we can use the algorithm described in Chapelle and Keerthi [12] to solve the linear ranking problem in $O(mm_+ + m \log m) = O(mm_+)$ time. Including the cost of factoring K_{++}, the total time is $O(mm_+ + m_+^3)$. Once we solve for optimal $\mathbf{w}$ we can use (15) to obtain $\boldsymbol{\beta}$ for subsequent testing purposes. Also, since we only need entries $\{K_{ij} : i = 1, ..., m, y_j = 1\}$, the method only requires $O(mm_+)$ space.

5.2. Unconstrained Optimization

We can also directly solve (14) 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 $\boldsymbol{\beta}^{(p)}$:

$$\min_{\mathbf{s}} \quad \mathbf{s}^T \mathbf{g}^{(p)} + \frac{1}{2} \mathbf{s}^T H^{(p)} \mathbf{s} \,, \tag{17}$$

where $H^{(p)}$ and $\mathbf{g}^{(p)}$ are the Hessian and gradient of the objective at $\boldsymbol{\beta}^{(p)}$, respectively. Problem (17) has a closed form solution given by

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

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 can be sensitive to the initial point. 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 Newton's method.

The idea behind the trust-region method is to solve (17) while constraining the step, $\mathbf{s}$, to a neighborhood around the current iterate, in which the approximation is trusted:

$$\begin{aligned} \min_{\mathbf{s}} \quad & \frac{1}{2} \mathbf{s}^T H^{(p)} \mathbf{s} + \mathbf{s}^T \mathbf{g}^{(p)} \\ \text{s.t.} \quad & ||\mathbf{s}||_2 \leq \Delta^{(p)} \,. \end{aligned} \tag{18}$$

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 (18) can be solved accurately [e.g see 29], however, the solution uses the full eigen-decomposition of $H^{(p)}$. To avoid this computation, in the subspace-trust-region method, Problem (18) 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 [30]. The idea behind this choice is to ensure global convergence (via steepest descent direction) and achieve fast local convergence (via the Newton step). Once the subspace has been computed, solving (18) 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 of Problem (14) efficiently. Let $K_{\bullet+} = [K_{ij}]_{i=1,\ldots,m,y_j=-1} \in \mathbb{R}^{m\times m_+}$ 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 $K_{\bullet+}$ 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:

$$A = DK_{\bullet+},$$

where $D \in \mathbb{R}^{m_+m_-\times m}$ 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 $D_{ri} = 1, D_{rj} = -1$ and the rest of the row is zero. Let A_r denote the rth row of A. Then the ranking loss expression in (14) can be written as,

$$\begin{aligned}
&\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \ell_\epsilon\left(\boldsymbol{\beta}^T K_{i+} - \boldsymbol{\beta}^T K_{j+}\right) \\
&= \sum_{r=1}^{m_+m_-} \ell_\epsilon\left(\boldsymbol{\beta}^T A_r\right) \\
&= \sum_{r=1}^{m_+m_-} \mathbb{I}[r\in\mathcal{L}]\left(1-\epsilon-\boldsymbol{\beta}^T A_r\right) + \sum_{r=1}^{m_+m_-} \mathbb{I}[r\in\mathcal{Q}]\frac{1}{4\epsilon}\left(1-\boldsymbol{\beta}^T A_r\right)^2, \qquad (19)
\end{aligned}$$

where $\mathcal{L} = \{r : \boldsymbol{\beta}^T A_r < 1-2\epsilon\}$ is the set of pairwise differences which are in the linear portion of ℓ_ϵ, and $\mathcal{Q} = \{r : 1-2\epsilon \le \boldsymbol{\beta}^T A_r < 1\}$ 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 (19) is equivalent to

$$\left(\mathbf{e}^{\mathcal{L}}\right)^T((1-\epsilon)\mathbf{e}-A\boldsymbol{\beta})+\frac{1}{4\epsilon}(\mathbf{e}-A\boldsymbol{\beta})^T I^{\mathcal{Q}}(\mathbf{e}-A\boldsymbol{\beta})$$
$$=\left(\mathbf{e}^{\mathcal{L}}\right)^T((1-\epsilon)\mathbf{e}-PK_{\bullet+}\boldsymbol{\beta})+\frac{1}{4\epsilon}(\mathbf{e}-PK_{\bullet+}\boldsymbol{\beta})^T I^{\mathcal{Q}}(\mathbf{e}-PK_{\bullet+}\boldsymbol{\beta})\ .$$

Therefore the objective function in (14) can be expressed as

$$F(\boldsymbol{\beta})\triangleq\frac{1}{m_+m_-}\left[\left(\mathbf{e}^{\mathcal{L}}\right)^T((1-\epsilon)\mathbf{e}-PK_{\bullet+}\boldsymbol{\beta})+\frac{1}{4\epsilon}(\mathbf{e}-PK_{\bullet+}\boldsymbol{\beta})^T I^{\mathcal{Q}}(\mathbf{e}-PK_{\bullet+}\boldsymbol{\beta})\right]+\frac{\lambda}{2}\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta}\ . \quad (20)$$

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

$$\begin{aligned}\mathbf{g}\triangleq\frac{\partial F}{\partial\boldsymbol{\beta}}&=\frac{1}{m_+m_-}\left[-\left(\mathbf{e}^{\mathcal{L}}\right)^T PK_{\bullet+}+\frac{1}{2\epsilon}PK_{\bullet+}I^{\mathcal{Q}}(PK_{\bullet+}\boldsymbol{\beta}-\mathbf{e})\right]+\lambda K_{++}\boldsymbol{\beta}\\&=\frac{1}{m_+m_-}\left[-\left(\left(\mathbf{e}^{\mathcal{L}}\right)^T P\right)K_{\bullet+}+\frac{1}{2\epsilon}\left(P\left(K_{\bullet+}\left(I^{\mathcal{Q}}P\right)(K_{\bullet+}\boldsymbol{\beta})\right)-P\left(K_{\bullet+}\left(I^{\mathcal{Q}}\mathbf{e}\right)\right)\right)\right]+\lambda K_{++}\boldsymbol{\beta}\ .\end{aligned} \quad (21)$$

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

We obtain the Hessian by taking the derivative of (21) with respect to $\boldsymbol{\beta}$:

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

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:

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

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. We found the solution usually converges in a constant number of trust region iterations. Since each iteration requires $O(mm_+)$ time, the total time required by the algorithm is $O(mm_+)$. 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 $K_{\bullet+}\boldsymbol{\beta}$ or $K_{\bullet+}\mathbf{s}$. 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 (14).

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 [23], in which the primal problem,

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

   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 [23, 31] in which

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

   is solved, with different weights associated with each class:

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

   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 RankSVM problem constrained to m_+ randomly selected set of basis function, i.e. Problem (**??**).
8. RANK-RC: We solve RankSVM constrained to the rare class representation, i.e. Problem (14).

We use LIBSVM [32] to solve the SVM problems (2-5). LIBSVM is a popular and efficient implementation of the sequential minimal optimization algorithm [33]. We set cache size to 10GB 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 10GB. Experiments are performed on a Xeon E5620@2.4Ghz 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 $\log_2\lambda = [-20,-18,...,8,10]$. For KNN we cross-validate over $k = [1,2,...,\lceil\min(100,\sqrt{m})\rceil]$.

6.1. Simulated Data

We simulate an unbalanced dataset in the following manner. Rare class instances are sampled from six multivariate normal distributions with equal probability. Their centers, μ_i, $i = 1,...,6$, 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 2-dimensional space is shown in Figure 4. The resulting dataset contains multiple rare-class subconcepts that vary in discriminative structure.

For our experiment we generate data in 5-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% in 10% increments for each t value. Thus we have a total of 12 datasets. For each dataset we generate 1000 training points, 1000 validation points and 10000 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.

Figure 5 compares the empirical ranking loss function,

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

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	**61.5±0.3**	**61.4±0.3**	**61.4±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	**61.6±0.3**	**61.3±0.4**	**62.3±0.3**	
	30%	57.3±1.0	61.1±0.2	**62.2±0.4**	62.2±0.1	61.8±0.3	**62.3±0.4**	**62.2±0.3**	**62.6±0.4**	
	40%	59.0±0.7	**62.8±0.2**	**63.2±0.3**	**63.3±0.2**	**62.8±0.2**	**62.7±0.3**	62.5±0.2	**62.7±0.3**	
Medium	10%	54.9±0.6	56.8±0.5	59.5±0.5	58.8±0.3	58.6±0.9	**61.4±0.5**	60.1±0.5	**61.4±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	**62.8±0.4**	61.5±0.4	**63.4±0.2**	
	30%	57.6±0.8	63.4±0.1	**64.0±0.3**	63.3±0.1	63.4±0.4	**64.1±0.4**	62.4±0.5	**64.6±0.5**	
	40%	58.0±0.7	**65.3±0.2**	**65.5±0.1**	**65.4±0.1**	**65.3±0.2**	64.6±0.3	63.0±0.2	**64.9±0.3**	
Low	10%	55.9±1.0	61.1±0.6	64.0±0.4	63.5±0.2	63.0±0.8	**65.3±0.5**	62.7±0.6	**65.5±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	**67.1±0.3**	63.5±0.4	**67.3±0.2**	
	30%	59.9±0.9	**69.1±0.4**	**69.4±0.2**	68.9±0.1	**69.2±0.2**	**69.2±0.4**	65.8±0.4	**69.8±0.4**	
	40%	61.6±1.5	**71.1±0.1**	**70.7±0.3**	70.5±0.1	**71.1±0.1**	69.8±0.3	65.8±0.3	69.9±0.3	

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 ρ =10%, 20%, 30% and 40% minority samples. 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% confidence.

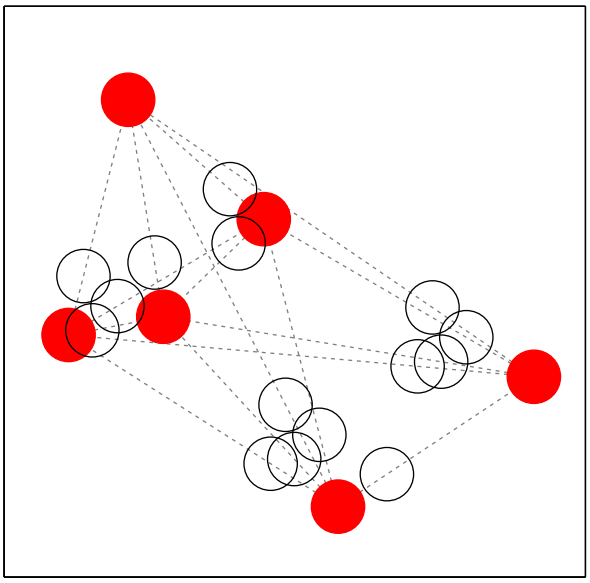


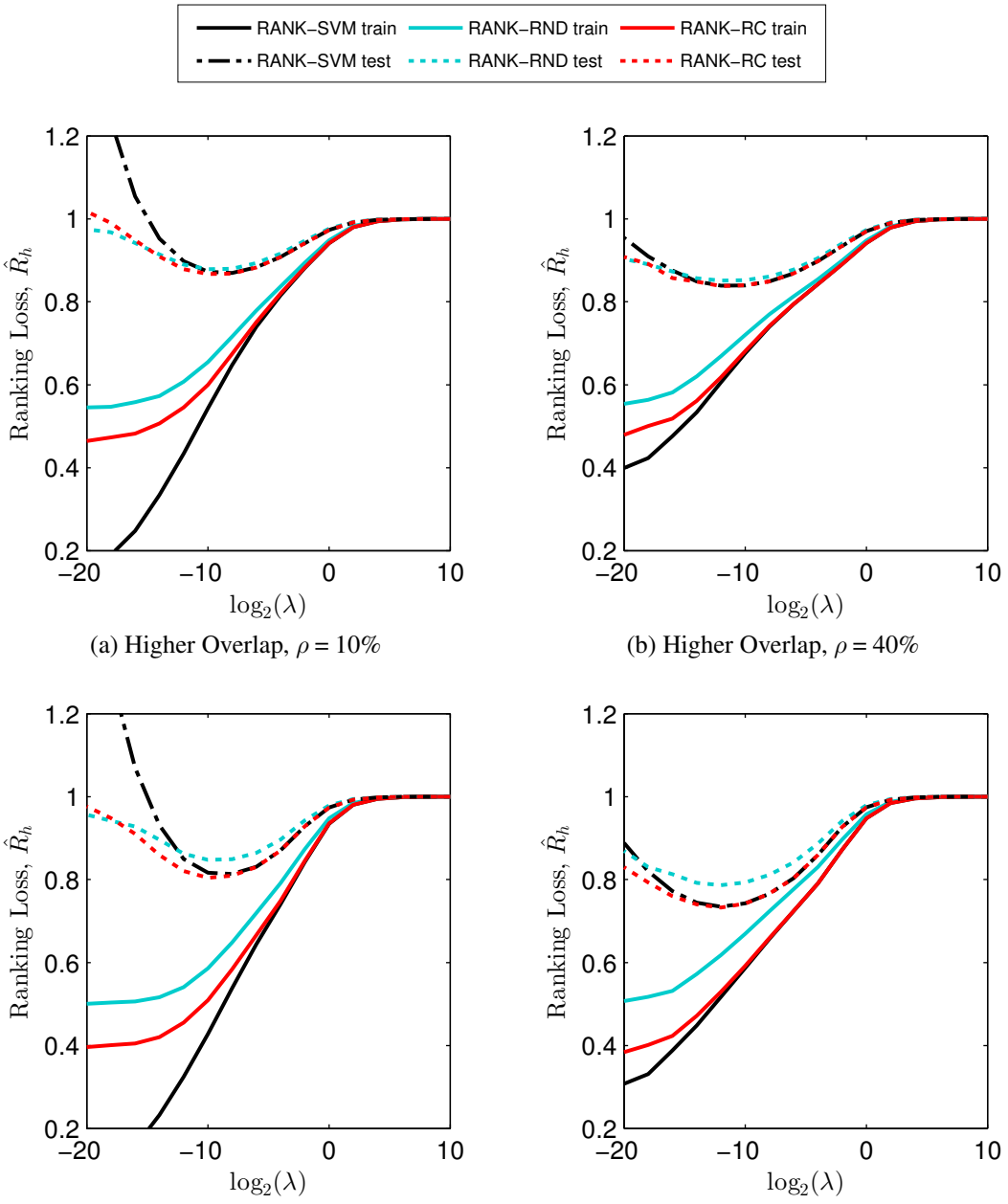
Figure 4: Example configuration of simulated dataset in 2-dimensions with $\sigma = 0.1$ and $t = 0.75$. The red, filled in circles show the locations of the six normal components for the rare class distribution. The black, empty circles show the location of the 15 normal components for the majority class distribution, whose centers lie along the dotted lines connecting all two rare class normal components.

obtained by the ranking methods on four of the training and testing sets as λ is varied. We observe that the difference between RANK-SVM and the restricted basis models (RANK-RND and RANK-RC) decreases as λ is increased. Since restricting basis functions also limits the complexity of the model, the test loss of RANK-RND and RANK-RC is lower than that of RANK-SVM for small values of λ. However, RANK-RND is unable to achieve the optimal test loss levels at moderate values of λ (more noticeably in Figures 5c and 5d). In contrast, RANK-RC does not forfeit any test performance compared to RANK-SVM, while providing additional robustness as λ is reduced.

6.2. Real Datasets

In this section we compare methods 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) Higher Overlap, $\rho = 10\%$

(b) Higher Overlap, $\rho = 40\%$

(c) Lower Overlap, $\rho = 10\%$

(d) Lower Overlap, $\rho = 40\%$

Figure 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 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 [24] for standard SVMs), would still run costly in time and space. In comparison, RANK-RND and RANK-RC use significantly fewer support vectors. Moreover, with RANK-RC, test performance is also not compromised.

6.3. Intrusion Detection

In this section we use the KDD Cup 1999 dataset [36] 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 311 029 test cases. We processed the data to remove duplicate entries (as done in [37]) 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 (3 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 4.0% 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 [38]. 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% positive instances.

We train using 5%, 10%, 25%, 50%, and 75% of the training data. The remaining training data is used for validation. We are unable to train RANK-SVM, even with just 5% of the data (53 749 samples), since the kernel matrix is too large to store in memory (>10GB). 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% of the data. With more samples SVM-W does not converge, likely due to the large number of support vectors which do not fit in the cache.

Figure 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. Figures 6b and 6c compare training time and number of support vectors, respectively, as training data is increased. SVM and SVM-RUS train in

Name	Source	Subject	Features		Samples		
			Original	Derived (d)	m	m_+	ρ
Abalone19	UCI	Life	1N,7C	10	4177	32	0.8%
Mammograph	[34]	Life	6C	6	11183	260	2.3%
Ozone	UCI	Environment	72C	72	2536	73	2.9%
YeastME2	UCI	Life	8C	8	1484	51	3.4%
Wine4	UCI	Chemistry	11C	11	4898	183	3.7%
Oil	[35]	Environment	49C	49	937	41	4.4%
SolarM0	UCI	Nature	10N	32	1389	68	4.9%
Coil	KDD	Business	85C	85	9822	586	6.0%
Thyroid	UCI	Life	21N,7C	52	3772	231	6.1%
Libras	UCI	Physics	90C	90	360	24	6.7%
Scene	LibSVM	Nature	294C	294	2407	177	7.4%
YeastML8	LibSVM	Life	103C	103	2417	178	7.4%
Crime	UCI	Economics	122C	100	1994	150	7.5%
Vowel0	Keel	Computer	10C	10	989	90	9.1%
Euthyroid	UCI	Life	18N,7C	42	3163	293	9.3%
Abalone7	UCI	Life	1N,7C	10	4177	391	9.4%
Satellite	UCI	Nature	36C	36	6435	626	9.7%
Page0	Keel	Computer	10C	10	5472	559	10.2%
Ecoli	UCI	Life	7C	7	336	35	10.4%
Contra2	Keel	Life	9C	9	1473	333	22.6%

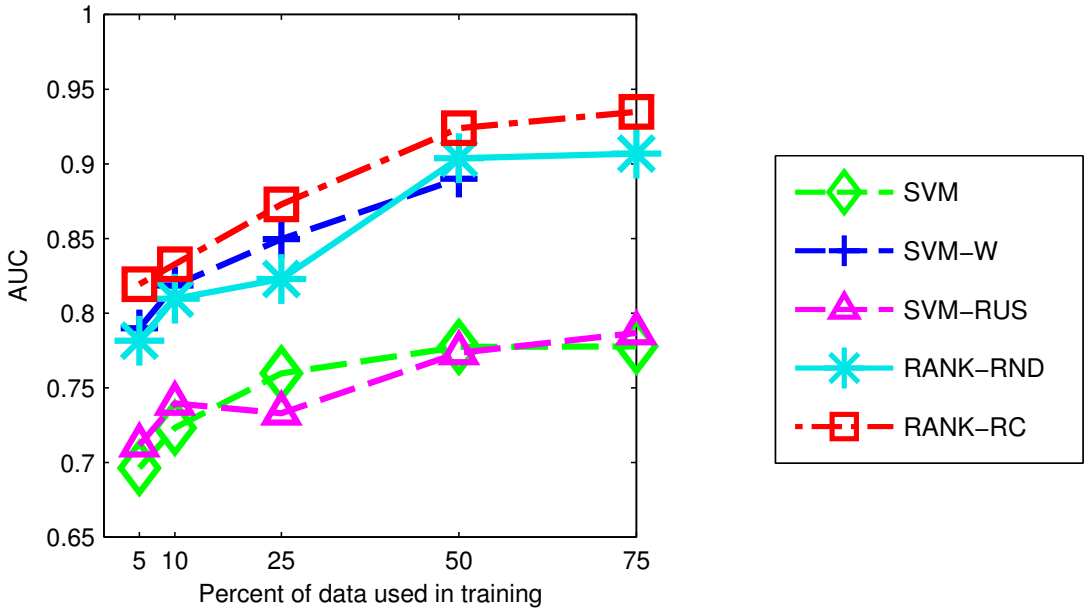
Table 3: List of datasets and their characteristics that we use to evaluate methods. Under original features, 'N' is used to denote number of nominal features, 'C', is used to denote number of continuous features. We derive d features by converting nominal features to an indicator representation and 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.

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

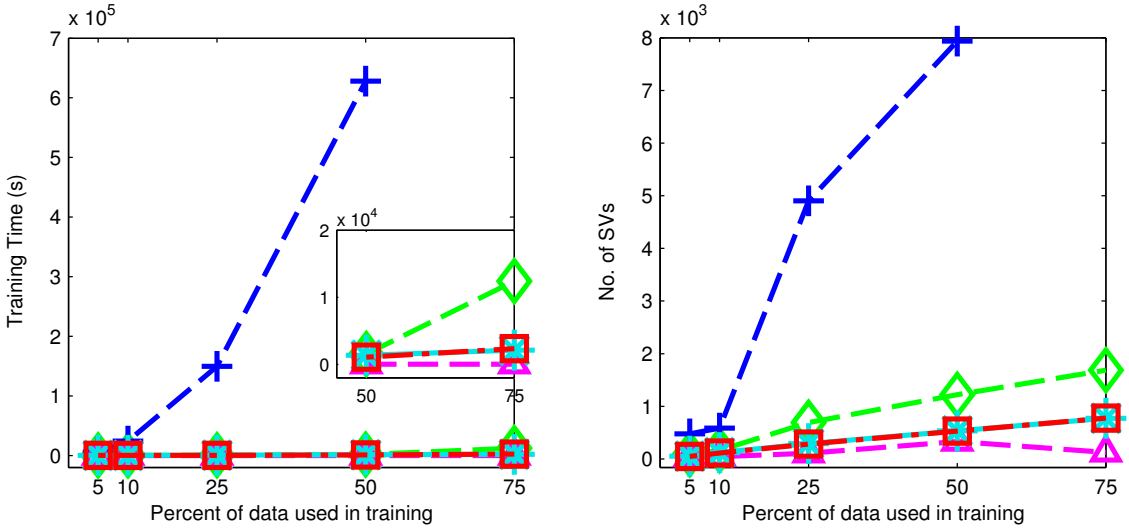
Table 4: Comparison of test AUC results for real datasets (listed in Table 3). Mean AUC score with standard error over 20 trials are shown. Each trial uses one-quarter data for out-of-sample testing. Bolded scores indicate the result is statistically not different than the best performing model using a two-tailed t-test with 99% confidence.

Dataset	Classification Loss				Ranking Loss		
	SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC
Abalone19	117	1555	32	2644	2979	24	24
Mammograph	306	2152	119	2987	7252	195	193
Ozone	206	746	61	1165	995	55	55
YeastME2	105	429	41	594	1066	38	38
Wine4	458	2109	179	3166	3550	136	136
Oil	84	311	32	340	488	31	31
SolarM0	183	845	90	1114	1042	51	51
Coil	1754	5560	704	8178	7284	435	435
Thyroid	332	723	137	1020	2739	168	172
Libras	35	93	22	124	197	18	18
Scene	603	1171	213	1566	1748	132	133
YeastML8	833	1669	257	1562	1804	133	133
Crime	308	631	115	867	1326	112	112
Vowel0	35	37	25	45	730	68	67
Euthyroid	389	673	177	1002	2303	216	219
Abalone7	713	1391	274	2076	3079	291	292
Satellite	773	1158	301	1526	4734	466	469
Page0	322	570	145	924	4012	415	416
Ecoli	47	68	18	115	248	26	26
Contra2	560	843	396	912	1096	249	249

Table 5: Average number of support vectors used by the SVM and ranking models over 20 trials. For ranking models, support vectors are counted as the number of non-zero coefficients associated with kernel functions.



(a)



(b) (c)

Figure 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%. In our experiment setup, we were unable to train RANK-SVM due to the large size of the dataset. Also, for more than 50% of data, SVM-W did not converge after more than 72 hours of training.

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). This allows us to solve the nonlinear ranking problem in $O(mm_+)$ time and $O(mm_+)$ space, thus enabling us to solve problems which are too large for kernel RankSVM. We provided heuristic and theoretical justification for this choice and experimentally illustrated the effectiveness of RankRC, in both test performance and training time.

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

1. Regularization: In problem (14) we can use an ℓ_1-regularizer, $\|\boldsymbol{\beta}\|_1$, instead of $\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta}$. This would lead to sparser solutions [39] and could be solved using coordinate descent methods [40].
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 [41] or the p-norm push [42] 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 [43]. 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 [44]. 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.

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

References

[1] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intell. Data Anal. 6 (2002) 429–449.

[2] B. Raskutti, A. Kowalczyk, Extreme re-balancing for svms: a case study, SIGKDD Explor. Newsl. 6 (2004) 60–69.

[3] G. Wu, E. Y. Chang, Class-boundary alignment for imbalanced dataset learning, in: ICML, 2003, pp. 49–56.

[4] G. E. A. P. A. Batista, R. C. Prati, M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explor. Newsl. 6 (2004) 20–29.
[5] N. V. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, SIGKDD Explor. Newsl. 6 (2004) 1–6.
[6] G. M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explor. Newsl. 6 (2004) 7–19.
[7] K. Ezawa, M. Singh, S. W. Norton, Learning goal oriented bayesian networks for telecommunications risk management, in: ICML, 1996, pp. 139–147.
[8] J. Zhang, I. Mani, KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction, in: ICML, 2003.
[9] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: One-sided selection, in: ICML, 1997, pp. 179–186.
[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.
[11] R. Herbrich, T. Graepel, K. Obermayer, Large Margin Rank Boundaries for Ordinal Regression, MIT Press, 2000.
[12] O. Chapelle, S. S. Keerthi, Efficient algorithms for ranking with svms, Inf. Retr. 13 (2010) 201–215.
[13] F. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms, in: In Proceedings of the Fifteenth International Conference on Machine Learning, 1997, pp. 445–453.
[14] M. A. Maloof, Learning when data sets are imbalanced and when costs are unequal and unknown, in: ICML, 2003.
[15] Y. Sun, M. S. Kamel, A. K. C. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recogn. 40 (2007) 3358–3378.
[16] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Trans. on Knowl. and Data Eng. 21 (2009) 1263–1284.
[17] A. P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (1997) 1145–1159.
[18] C. E. Metz, Basic principles of ROC analysis., Seminars in nuclear medicine 8 (1978) 283–298.
[19] J. A. Hanley, B. J. Mcneil, The meaning and use of the area under a receiver operating characteristic (ROC) curve., Radiology 143 (1982) 29–36.
[20] E. R. DeLong, D. M. DeLong, D. L. Clarke-Pearson, Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach, Biometrics 44 (1988) 837–845.
[21] P. L. Bartlett, M. I. Jordan, J. D. McAuliffe, Convexity, Classification, and Risk Bounds, Journal of the American Statistical Association 101 (2006) 138–156.
[22] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, COLT '92, ACM, 1992, pp. 144–152.
[23] V. N. Vapnik, Statistical Learning Theory, 1st ed., Wiley, 1998.
[24] O. Chapelle, Training a support vector machine in the primal, Neural Comput. 19 (2007) 1155–1178.
[25] G. Kimeldorf, G. Wahba, A correspondence between Bayesian estimation of stochastic processes and smoothing by splines, Ann. Math. Statist. 41 (1970) 495–502.
[26] B. Schölkopf, R. Herbrich, A. J. Smola, A generalized representer theorem, in: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory, 2001, pp. 416–426.
[27] T. Joachims, Optimizing search engines using clickthrough data, in: KDD, 2002, pp. 133–142.
[28] M. Zhu, W. Su, H. A. Chipman, LAGO: A Computationally Efficient Approach for Statistical Detection, Technometrics 48 (2006).
[29] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[30] M. A. Branch, T. F. Coleman, Y. Li, A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems, SIAM J. Scientific Computing 21 (1999) 1–23.
[31] E. E. Osuna, R. Freund, F. Girosi, Support Vector Machines: Training and Applications, Technical Report, MIT, 1997.
[32] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Trans-

actions on Intelligent Systems and Technology 2 (2011) 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[33] J. C. Platt, Advances in kernel methods, MIT Press, 1999, pp. 185–208.
[34] K. Woods, C. Doss, K. Bowyer, J. Solka, C. Preibe, P. Keglmyer, Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography, International Journal of Pattern Recognition and Artificial Intelligence 7 (1993) 1417–1436.
[35] M. Kubat, R. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Machine Learning 30 (1998) 195–215.
[36] KDD Cup 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Accessed: 2013-08-31.
[37] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications, CISDA'09, 2009, pp. 53–58.
[38] M. Sabhnani, Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context, in: ICML, 2003, pp. 209–215.
[39] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Roy. Statist. Soc. Ser. B 58 (1996) 267–288.
[40] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software (2009).
[41] C. G. Weng, J. Poon, A new evaluation measure for imbalanced datasets, in: Seventh Australasian Data Mining Conference (AusDM 2008), volume 87, 2008, pp. 27–32.
[42] C. Rudin, The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list, J. Mach. Learn. Res. 10 (2009) 2233–2271.
[43] T. Joachims, A support vector method for multivariate performance measures, in: ICML, 2005, pp. 377–384.
[44] L. Bottou, O. Bousquet, The tradeoffs of large scale learning, in: NIPS, 2008, pp. 161–168.

	Training		Test	
Normal	812808	75.6%	47913	62.0%
DOS	247266	23.0%	23568	30.5%
Probing	13850	1.3%	2677	3.5%
U2R	52	0.005%	215	0.278%
R2L	999	0.093%	2913	3.769%
Total	1074975	100%	77286	100%

Table 6: Distribution of connection types in training and test sets for the intrusion detection problem.