The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

τ-FPL: Tolerance-Constrained Learning in Linear Time

Ao Zhang,[1] **Nan Li,**[2] **Jian Pu,**[1] **Jun Wang,**[1] **Junchi Yan,**[31] **Hongyuan Zha**[14]
[1]Shanghai Key Laboratory of Trustworthy Computing, MOE International Joint Lab of Trustworthy Software,
School of Computer Science and Software Engineering, East China Normal University, Shanghai, China
[2]Institute of Data Science and Technologies, Alibaba Group, Hangzhou, China
[3]IBM Research – China [4]Georgia Institute of Technology, Atlante, USA
az.aozhang@gmail.com, nanli.ln@alibaba-inc.com, {jianpu,jwang,zha}@sei.ecnu.edu.cn, yanesta@163.com

Abstract

In many real-world applications, learning a classifier with false-positive rate under a specified tolerance is appealing. Existing approaches either introduce prior knowledge dependent label cost or tune parameters based on traditional classifiers, which are of limitation in methodology since they do not directly incorporate the false-positive rate tolerance. In this paper, we propose a novel scoring-thresholding approach, τ-*False Positive Learning* (τ-FPL) to address this problem. We show that the scoring problem which takes the false-positive rate tolerance into accounts can be efficiently solved in linear time, also an out-of-bootstrap thresholding method can transform the learned ranking function into a low false-positive classifier. Both theoretical analysis and experimental results show superior performance of the proposed τ-FPL over the existing approaches.

Introduction

In real-world applications, such as spam filtering (Drucker, Wu, and Vapnik 1999) and medical diagnosing (Huang, Liu, and Zhou 2010), the loss of misclassifying a positive instance and negative instance can be rather different. For instance, in medical diagnosing, misdiagnosing a patient as healthy is more dangerous than misclassifying healthy person as sick. Meanwhile, in reality, it is often infeasible to define an accurate cost for these two kinds of errors (Liu and Zhou 2010; Zhou and Zhou 2016). In such situations, it is often needed to keep the classifier working under a small tolerance of false-positive rate (FPR) τ, i.e., only allow the classifier to misclassify no larger than τ percent of negative instances. Traditional classifiers trained by maximizing classification accuracy or AUC are not suitable due to mismatched goal.

In the literature, classification under constrained false-positive rate is known as Neyman-Pearson (NP) Classification problem (Scott and Nowak 2005; Lehmann and Romano 2006; Rigollet and Tong 2011), and existing approaches can be roughly grouped into several categories. One common approach is to use *cost-sensitive learning*, which assigns different costs for different classes, and representatives include cost-sensitive SVM (Osuna, Freund, and Girosi 1997; Davenport, Baraniuk, and Scott 2006; 2010), cost-interval SVM (Liu and Zhou 2010) and cost-sensitive boosting (Masnadi-Shirazi and Vasconcelos 2007; 2011). Though effective and efficient in handling different misclassification costs, it is usually not easy to find appropriate misclassification cost for specific FPR tolerance. Another group of methods formulates this problem as a constrained optimization problem, which has the FPR tolerance as an explicit constraint (Mozer et al. 2002; Gasso et al. 2011; Mahdavi, Yang, and Jin 2013). These methods often need to find the saddle point of Lagrange function, leading to time-consuming alternate optimization. Moreover, a surrogate loss is often used to simplify the optimization problem, possibly making the tolerance constraint not satisfied in practice. The third line of research is scoring-thresholding methods, which train a scoring function first, then find a threshold to meet the target FPR tolerance (Drucker, Wu, and Vapnik 1999). In practice, the scoring function can be trained by either class conditional density estimation (Tong 2013) or bipartite ranking (Narasimhan and Agarwal 2013a). However, density estimation itself is another difficulty. Also most bipartite ranking methods have super-linear training complexity, which limits their scalability. Meanwhile, there are some methods paying special attention to the positive class. For example, asymmetric SVM (Wu et al. 2008) maximizes the margin between negative samples and the core of positive samples, one-class SVM (Ben-Hur et al. 2001) finds the smallest ball to enclose positive samples. However, they do not incorporate the FPR tolerance into the learning procedure either.

In this paper, we address the tolerance constrained learning problem by proposing τ-*False Positive Learning* (τ-FPL). Specifically, τ-FPL is a scoring-thresholding method. In the scoring stage, we explicitly learn a ranking function which optimizes the probability of ranking any positive instance above the centroid of the worst τ percent of negative instances. Whereafter, it is shown that, with the help of our newly proposed Euclidean projection algorithm, this ranking problem can be solved in linear time under the projected gradient framework. It is worth noting that the Euclidean projection problem is a generalization of a large family of projection problems, and our proposed linear-time algorithm based on bisection and divide-and-conquer is one to three orders faster than existing state-of-the-art methods. In the thresholding stage, we devise an out-of-bootstrap threshold-

ing method to transform aforementioned ranking function into a low false-positive classifier, which is more stable compared to existing thresholding method. Theoretical analysis and experimental results show that the proposed method achieves superior performance over existing approaches.

From Constrained Optimization to Ranking

In this section, we show that the FPR tolerance problem can be transformed into a ranking problem, then give a convex ranking loss function.

Let $\mathcal{X} = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^d : ||\boldsymbol{x}|| \leq 1\}$ be the instance space, and $\mathcal{S} = \mathcal{S}_+ \cup \mathcal{S}_-$ be a set of training instances, where $\mathcal{S}_+ = \{\boldsymbol{x}_i^+ \in \mathcal{X}\}_{i=1}^m$ and $\mathcal{S}_- = \{\boldsymbol{x}_j^- \in \mathcal{X}\}_{j=1}^n$ contains m and n instances independently sampled from distributions $\mathbb{P}^+$ and $\mathbb{P}^-$, respectively. Let $0 \leq \tau \ll 1$ be the maximum tolerance of false-positive rate. Consider the following optimization problem with false-positive rate constraint, which is known as Neyman-Pearson classification criteria[1]:

$$\begin{aligned} \min_{f,b} \quad & \mathbb{P}_{\boldsymbol{x}^+ \sim \mathbb{P}^+}(f(\boldsymbol{x}^+) < b) \\ \text{s.t.} \quad & \mathbb{P}_{\boldsymbol{x}^- \sim \mathbb{P}^-}(f(\boldsymbol{x}^-) > b) \leq \tau \end{aligned} \tag{1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is a scoring function and $b \in \mathbb{R}$ a threshold. With finite training instances, the corresponding *empirical risk minimization* problem is

$$\begin{aligned} \min_{f,b} \quad & \mathcal{L}_{emp}(f,b) = \frac{1}{m}\sum\nolimits_{i=1}^{m} \mathbb{I}(f(\boldsymbol{x_i^+}) < b) \\ \text{s.t.} \quad & \frac{1}{n}\sum\nolimits_{j=1}^{n} \mathbb{I}(f(\boldsymbol{x_j^-}) > b) \leq \tau \end{aligned} \tag{2}$$

where $\mathbb{I}(u)$ is the indicator function. Although directly optimizing by finding a saddle point of Lagrangian function is available, it usually falls into a time consuming alternate optimization framework (Mozer et al. 2002; Gasso et al. 2011). Indeed, there have been considerable efforts on approximating this problem by introducing asymmetric costs for different type of error into classification learning framework (Davenport, Baraniuk, and Scott 2010)

$$\min_{f,b} \mathcal{L}_{emp}^C(f,b) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(f(\boldsymbol{x_i^+}) < b) + \frac{C}{n}\sum_{j=1}^{n} \mathbb{I}(f(\boldsymbol{x_j^-}) > b)$$

where $C \geq 0$ is a hyper-parameter that punishes the gain of false positive instance. Although reasonable, here we point out that, all methods under this framework indeed *minimize a lower bound* of problem (2). This can be verified by formulating (2) into unconstrained form $\mathcal{L}'_{emp}(f,b)$

$$\begin{aligned} & \mathcal{L}'_{emp}(f,b) \\ \triangleq \quad & \max_{\lambda \geq 0} \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(f(\boldsymbol{x_i^+}) < b) + \lambda\left(\frac{1}{n}\sum_{j=1}^{n} \mathbb{I}(f(\boldsymbol{x_j^-}) > b) - \tau\right) \\ \geq \quad & \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(f(\boldsymbol{x_i^+}) < b) + C\left(\frac{1}{n}\sum_{j=1}^{n} \mathbb{I}(f(\boldsymbol{x_j^-}) > b) - \tau\right) \\ = \quad & \mathcal{L}_{emp}^C(f,b) - C\tau. \end{aligned}$$

Thus, for a fixed C, minimizing $\mathcal{L}_{emp}^C$ is equivalent to minimize a lower bound of $\mathcal{L}_{emp}$. In other words, cost-sensitive learning methods are *insecure* in this setting.

[1]The case of $f(\boldsymbol{x}) = b$ is omit for simplicity .

On the other hand, very few work tries to minimize a more reasonable upper bound of (2) or its equivalent problems. It is also lack of clear formulation for such upper bound. We address these problems in detail below.

Proposition 1. *Denote $f(\boldsymbol{x}_{[j]}^-)$ the j-th largest value in set $\{f(\boldsymbol{x}_i) \mid \boldsymbol{x}_i \in \mathcal{S}_-\}$, and $\lceil . \rceil$ the ceil function. Then the constrained optimization problem (2) share the same optimal solution f^* with following ranking problem*

$$\min_f \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\left(f(\boldsymbol{x}_i^+) - f(\boldsymbol{x}_{[\lceil \tau n \rceil]}^-) < 0\right). \tag{3}$$

Proof. For a fixed f, it is clear that the constraint in (2) is equivalent to $b \geq f(\boldsymbol{x}_{[\lceil \tau n \rceil]}^-)$. Since the objective in (2) is a non-increasing function of b, its minimum is achieved at $b = f(\boldsymbol{x}_{[\lceil \tau n \rceil]}^-)$. From this, we can transform the original problem (2) into its equivalent form (3) by substituting. □

Proposition 1 constructs the connection between constrained optimization (2) and ranking problem (3). Optimizing (3) is very difficult due to the participation of operation $[\cdot]$, which is non-convex when $\lceil \tau n \rceil \geq 2$. Hence, we consider to optimize following upper bound of (3)

$$\min_f \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\left(f(\boldsymbol{x}_i^+) - \frac{1}{\lceil \tau n \rceil}\sum_{i=1}^{\lceil \tau n \rceil} f(\boldsymbol{x}_{[i]}^-) < 0\right). \tag{4}$$

which prefers scores on positive examples to exceed the mean of scores on the worst τ-proportion of negative examples. If $\tau = 0$, it is equivalent to the original problem (3). In general cases, equality also could hold when both the scores of the worst τ-proportion negative examples are the same.

The advantages of considering ranking problem (4) include: by replacing $\mathbb{I}(\cdot)$ by its convex surrogate, it produces a tight convex upper bound of the original minimization problem, in contrast to cost-sensitive classification which may only offer an unstable lower bound; its formulation leads to a *linear time* ranking algorithm, which remains the same training time complexity compared with cost-sensitive classification, and outperforms most of the traditional ranking algorithms; It is also cost-free, and the generalization performance is not depending on additional hyper-parameters.

Tolerance-Constrained False Positive Learning

Based on previous discussion, our framework can be divided into two stages, namely *scoring* and *thresholding*. In scoring, a function $f(\cdot)$ is learnt to maximize the probability of giving higher a score to positive instances than the centroid of top τ percent of the negative instances. In thresholding, a suitable threshold b will be chosen, and the final prediction of an instance $\boldsymbol{x}$ can be obtained by

$$y = \operatorname{sgn}(f(\boldsymbol{x}) - b)\ . \tag{5}$$

Tolerance-Constrained Ranking

In (4), we consider linear scoring function $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$, where $\boldsymbol{w} \in \mathbb{R}^d$ is the weight vector to be learned, and replace $\mathbb{I}(u < 0)$ by its convex surrogate function $l(u) = [1-u]_+^2$. Here

Algorithm 1 τ-FPL Ranking

Require: $X^+ \in \mathbb{R}^{m\times d}, X^- \in \mathbb{R}^{n\times d}$, maximal FPR tolerance τ, regularization parameter R, stopping condition ϵ

1: Randomly initialize $\boldsymbol{\alpha_0}$ and $\boldsymbol{\beta_0}$
2: Set counter: $t \leftarrow 0$
3: **while** $t = 0$ **or** $|g(\boldsymbol{\alpha_t}, \boldsymbol{\beta_t}) - g(\boldsymbol{\alpha_{t-1}}, \boldsymbol{\beta_{t-1}})| > \epsilon$ **do**
4: Compute gradient of $g(\cdot)$ at point $(\boldsymbol{\alpha_t}, \boldsymbol{\beta_t})$
5: Compute $\boldsymbol{\alpha'_{t+1}}, \boldsymbol{\beta'_{t+1}}$ by gradient descent;
6: Project $\boldsymbol{\alpha'_{t+1}}, \boldsymbol{\beta'_{t+1}}$ onto the feasible set Γ_k:

$$(\boldsymbol{\alpha_{t+1}}, \boldsymbol{\beta_{t+1}}) \leftarrow \Pi_{\Gamma_k}(\boldsymbol{\alpha'_{t+1}}, \boldsymbol{\beta'_{t+1}})$$

7: Update counter: $t \leftarrow t + 1$;
8: **end while**
9: Return $\boldsymbol{w} \leftarrow (mR)^{-1}(\boldsymbol{\alpha_t}^T X^+ - \boldsymbol{\beta_t}^T X^-)^T$

$[x]_+ \triangleq \max\{x, 0\}$. Kernel methods can be used for nonlinear ranking functions. As a result, the learning problem is

$$\min_{\boldsymbol{w}} \frac{1}{m}\sum_{i=1}^{m}\left[1 - \boldsymbol{w}^T\boldsymbol{x_i^+} + \frac{1}{k}\sum_{j=1}^{k}\boldsymbol{w}^T\boldsymbol{x}_{[j]}^-\right]_+^2 + \frac{R}{2}\|\boldsymbol{w}\|^2 \quad (6)$$

where $R > 0$ is the regularization parameter, and $k = \lceil \tau n \rceil$.

Directly minimizing (6) can be challenging due to the $[\cdot]$ operator, we address it by developing its dual.

Theorem 1. *Define $\boldsymbol{X}^+ = [\boldsymbol{x}_1^+, ..., \boldsymbol{x}_m^+]^T$ and $\boldsymbol{X}^- = [\boldsymbol{x}_1^-, ..., \boldsymbol{x}_m^-]^T$ be the matrix containing positive and negative instances in their rows respectively, the dual problem of (6) can be written by*

$$\min_{(\boldsymbol{\alpha},\boldsymbol{\beta}\in\Gamma_k)} g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2mR}||\boldsymbol{\alpha}^T\boldsymbol{X}^+ - \boldsymbol{\beta}^T\boldsymbol{X}^-||^2 + \sum_{i=1}^{m} l^*(-\alpha_i) \quad (7)$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are dual variables, $l^*(\cdot)$ is the convex conjugate of $l(\cdot)$, and the domain Γ_k is defined as

$$\Gamma_k = \left\{\boldsymbol{\alpha} \in \mathbb{R}_+^m, \boldsymbol{\beta} \in \mathbb{R}_+^n \mid \sum_{i=1}^{m}\alpha_i = \sum_{j=1}^{n}\beta_j; \beta_j \le \frac{1}{k}\sum_{i=1}^{m}\alpha_i, \forall j\right\}.$$

Let $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ be the optimal solution of (7), the optimal

$$\boldsymbol{w}^* = (mR)^{-1}(\boldsymbol{\alpha}^{*T}\boldsymbol{X}^+ - \boldsymbol{\beta}^{*T}\boldsymbol{X}^-)^T \quad (8)$$

Proof. Due to space limit, the proof is put in appendix. □

According to Theorem 1, learning scoring function f is equivalent to learning the dual variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by solving problem (7). Its optimization naturally falls into the area of projected gradient method. The key steps are summarized in Algorithm 1. At each iteration, we first update solution by the gradients of the objective function $g(\boldsymbol{\alpha}, \boldsymbol{\beta})$, then project the dual solution onto feasible set Γ_k. In the sequel, we will show that this projection problem can be efficiently solved in linear time. In practice, since $g(\cdot)$ is smooth, we also leverage Nesterov's method to further accelerate the convergence of our algorithm. Nesterov's method (Nesterov 2003) achieves $O(1/T^2)$ convergence rate for smooth objective function, where T is the number of iterations.

Linear Time Projection onto the Top-k Simplex

One of our main technical results is a *linear time* projection algorithm onto Γ_k, even in the case of $k \to n$. For clear notations, we reformulate the projection problem as

$$\min_{\boldsymbol{\alpha}\ge 0, \boldsymbol{\beta}\ge 0} \frac{1}{2}||\boldsymbol{\alpha} - \boldsymbol{\alpha}^0||^2 + \frac{1}{2}||\boldsymbol{\beta} - \boldsymbol{\beta}^0||^2 \quad (9)$$
$$\text{s.t. } \sum_{i=1}^{m}\alpha_i = \sum_{j=1}^{n}\beta_j, \quad \beta_j \le \frac{1}{k}\sum_{i=1}^{m}\alpha_i, \forall j.$$

It should be noted that, many Euclidean projection problems studied in literature can be seen as a special case of this problem. If the term $\sum_{i=1}^{m}\alpha_i$ is fixed, or replaced by a constant upper bound C, we obtain a well studied case of *continuous quadratic knapsack problem* (CQKP)

$$\min_{\boldsymbol{\beta}} ||\boldsymbol{\beta} - \boldsymbol{\beta}^0||^2 \quad \text{s.t. } \sum_{i=1}^{n}\beta_i \le C, 0 \le \beta_i \le C_1,$$

where $C_1 = C/k$. Several efficient methods based on median-selecting or variable fixing techniques are available (Patriksson 2008). On the other hand, if $k = 1$, all upper bounded constraints are automatically satisfied and can be omitted. Such special case has been well studied, for example, in (Liu and Ye 2009) and (Li, Jin, and Zhou 2014), both of which achieve $O(n)$ complexity.

Unfortunately, none of those above methods can be directly applied to solving the generalized case (9), due to its property of unfixed upper-bound constraint on β when $k > 1$. To our knowledge, the only one attempt to address the problem of unfixed upper bound is (Lapin, Hein, and Schiele 2015). They solve a similar (but simpler) problem

$$\min_{\beta} ||\boldsymbol{\beta} - \boldsymbol{\beta}^0||^2 \text{ s.t. } 0 \le \beta_j \le \frac{1}{k}\sum_{i=1}^{n}\beta_i$$

based on sorting and exhaustive search and their method achieves a runtime complexity $O(n\log(n) + kn)$, which is super-linear and even quadratic when k and n are linearly dependent. By contrast, our proposed method can be applied to both of aforementioned special cases with slight changes and remains $O(n)$ complexity. The notable characteristic of our method is the efficient combination of bisection and divide-and-conquer: the former offers the guarantee of worst complexity, and the latter significantly reduces the large constant factor of bisection method.

We first introduce the following theorem, which gives a detailed description of the solution for (9).

Theorem 2. *$(\boldsymbol{\alpha}^* \in \mathbb{R}^m, \boldsymbol{\beta}^* \in \mathbb{R}^n)$ is the optimal solution of (9) if and only if there exist dual variables $C^* \ge 0$, λ^*, $\mu^* \in \mathbb{R}$ satisfy the following system of linear constraints:*

$$C^* = \sum_{i=1}^{m}[\alpha_i^0 - \lambda^*]_+ \quad (10)$$
$$C^* = \sum_{j=1}^{n}\min\{[\beta_j^0 - \mu^*]_+, C^*/k\} \quad (11)$$
$$0 = \lambda^* + \mu^* + \frac{1}{k}\sum_{j=1}^{n}\left[\beta_j^0 - \mu^* - C^*/k\right]_+ \quad (12)$$

and $\alpha_i^* = [\alpha_i^0 - \lambda^*]_+$, $\beta_j^* = \min\{[\beta_j^0 - \mu^*]_+, C^*/k\}$.

Proof. Due to space limit, the proof is put in appendix. □

Based on Theorem 2, the projection problem can be solved by finding the value of three dual variables C, λ and μ

that satisfy the above linear system. Here we first propose a baseline bisection method which guarantees the worst time complexity. Similar method has also been used in (Liu and Ye 2009). For brevity, we denote $\alpha^0_{[i]}$ and $\beta^0_{[i]}$ the i-largest dimension in $\boldsymbol{\alpha^0}$ and $\boldsymbol{\beta^0}$ respectively, and define function $C(\lambda)$, $\mu(C)$, $\delta(C)$ and $f(\lambda)$ as follows[2]:

$$C(\lambda) = \sum_{i=1}^{m}[\alpha_i^0 - \lambda]_+ \quad (13)$$

$$\mu(C) = \mu \text{ satisfies } (11) \quad (14)$$

$$\delta(C) = \mu(C) + C/k \quad (15)$$

$$f(\lambda) = k\lambda + k\mu(C(\lambda)) + \sum_{j=1}^{n}[\beta_j^0 - \delta(C(\lambda))]_+ \quad (16)$$

The main idea of leveraging bisection to solve the system in theorem 2 is to find the root of $f(\lambda) = 0$. In order to make bisection works, we need three guarantees: f should be continuous; root of f should be efficiently bounded in a interval; the value of f at the two endpoints of this interval should have opposite signs. Fortunately, based on following three lemmas, both of these requirements can be ensured.

Lemma 1. *(Zero case)* $(\mathbf{0}^m, \mathbf{0}^n)$ *is an optimal solution of (9) if and only if* $k\alpha^0_{[1]} + \sum_{j=1}^{k}\beta^0_{[j]} \leq 0$.

Lemma 2. *(Bounding λ^*)* *If* $C^* > 0$, $\lambda^* \in (-\beta^0_{[1]},\ \alpha^0_{[1]})$.

Lemma 3. *(Monotonicity and convexity)*

1. $C(\lambda)$ *is convex, continuous and strictly decreasing in* $(-\infty,\ \alpha^0_{[1]})$*;*
2. $\mu(C)$ *is continuous, monotonically decreasing in* $(0, +\infty)$*;*
3. $\delta(C)$ *is continuous, strictly increasing in* $(0, +\infty)$*;*
4. $f(\lambda)$ *is continuous, strictly increasing in* $(-\infty, \alpha^0_{[1]})$.

Furthermore, we can define the inverse function of $C(\lambda)$ *as* $\lambda(C)$*, and rewrite* $f(\lambda)$ *as:*

$$f(\lambda(C)) = k\lambda(C) + k\mu(C) + \sum_{j=1}^{n}[\beta_j^0 - \delta(C)]_+ \quad (17)$$

it is a convex function of C*, strictly decreasing in* $(0, +\infty)$.

Lemma 1 deal with the special case of $C^* = 0$. Lemma 2 and 3 jointly ensure that bisection works when $C^* > 0$; Lemma 2 bounds λ^*; Lemma 3 shows that f is continuous, and since it is also strictly increasing, the value of f at two endpoints must has opposite sign.

Baseline method We start from select current λ in range $(-\beta^0_{[1]},\ \alpha^0_{[1]}) \triangleq [l, u]$. Then compute corresponding C by (13) in $O(m)$, and use current C to compute μ by (14). Computing μ can be completed in $O(n)$ by a well-designed median-selecting algorithm (Kiwiel 2007). With current (i.e. updated) C, λ and μ in hand, we can evaluate the sign of $f(\lambda)$ in $O(n)$ and determine the new bound of λ. In addition, the special case of $C = 0$ can be checked using Lemma 1 in $O(m+n)$ by a linear-time k-largest element selecting algorithm (Kiwiel 2005). Since the bound of λ is irrelevant to m and n, the number of iteration for finding λ^* is $\log(\frac{u-l}{\epsilon})$, where ϵ is the maximum tolerance of the error. Thus, the worst runtime of this algorithm is $O(m+n)$. Furthermore, we also leverage the convexity of $f(\lambda(C))$ and $C(\lambda)$ to further improve this algorithm, please refer to (Liu and Ye 2009) for more details about related technologies.

[2]Indeed, for some C, $\mu(C)$ is not one-valued and thus need more strict definition. Here we omit it for brevity, and leave details in supplement materials.

Algorithm 2 Linear-time Projection on Top-k simplex

Require: $\boldsymbol{\alpha^0} \in \mathbb{R}^m, \boldsymbol{\beta^0} \in \mathbb{R}^n$, maximal accuracy ϵ

1: Calculate initial uncertainly intervals for λ, C, δ and μ;
2: Initialize breakpoint caches for $C(\lambda)$, $\mu(C)$, $\delta(C)$, $f(\lambda)$:

$$Cache_C \leftarrow \{\alpha_i^0 \mid \forall i\},\ Cache_\mu, Cache_\delta, Cache_f \leftarrow \{\beta_j^0 \mid \forall j\}$$

3: Initialize partial sums of $C(\lambda)$, $\mu(C)$, $\delta(C)$, $f(\lambda)$ with zero;
4: Set $t \leftarrow 0$ and $\lambda_0 \leftarrow (\alpha^0_{[1]} - \beta^0_{[1]})/2$;
5: **while** $t = 0$ **or** $|\lambda_t - \lambda_{t-1}| > \epsilon$ **do**
6: Calculate C_t, μ_t, δ_t, f_t(by leveraging corresponding caches and partial sums);
7: Prune caches and update partial sums;
8: Shrink intervals of λ, C, δ and μ based on sign of $f(\lambda_t)$;
9: $t \leftarrow t + 1$;
10: Set λ_t as midpoint of current new interval
11: **end while**
12: Return $\lambda^* \leftarrow \lambda_t$, $\mu^* \leftarrow \mu_t$, $C^* \leftarrow C_t$

Although bisection solves the projections in linear time, it may lead to a slow convergence rate. We further improve runtime complexity by reducing the constant factor $\log(\frac{u-l}{\epsilon})$. This technology benefits from exploiting the monotonicity of both functions $C(\lambda)$, $\mu(C)$, $\delta(C)$ and $f(\lambda)$, which have been stated in Lemma 3.

Improved method by endpoints Divide & Conquer Lemma 3 reveals an important chain monotonicity between the dual variables, which can used to improve the performance of our baseline method. The key steps are summarized in Algorithm 2. Denote the value of a variable z in iteration t as z_t. For instance, if $\lambda_t > \lambda_{t-1}$, from emma 3 we have $C_t < C_{t-1}$, $\mu_t > \mu_{t-1}$ and $\delta_t < \delta_{t-1}$. This implies that we can set uncertainty intervals for both λ, C, μ and δ. As the interval of λ shrinking, lengths of these four intervals can be reduced simultaneously. On the other hand, notice that $C(\lambda)$ is indeed piecewise linear function (at most $m+1$ segments), the computation of its value only contains a comparison between λ_t and all of the α_i^0s. By keeping a cache of α_i^0s and discard those elements which are out of the current bound of λ in advance, in each iteration we can reduce the expected comparison counts by half. A more complex but similar procedure can also be applied for computing $\mu(C)$, $\delta(C)$, and $f(\lambda)$, because both of these functions are piecewise linear and the main cost is the comparison with $O(m+n)$ endpoints. As a result, for approximately linear function (convexity not required) and evenly distributed breakpoints, if the first iteration of bisection costs $\gamma(m+n)$ time, the overall runtime of the projection algorithm will be $\gamma(m+n) + \gamma(m+n)/2 + ... \leq 2\gamma(m+n)$, which is much less than the original bisection algorithm whose runtime is $\log(\frac{u-l}{\epsilon})\gamma(m+n)$.

Algorithm	Training	Validation
τ-FPL	$O(d(m+n)/T^2)$	Linear
TopPush	$O(d(m+n)/T^2)$	Linear
CS-SVM	$O(d(m+n)/T)$	Quadratic
$\text{SVM}^{\text{pAUC}}_{\text{tight}}$	$O((m\log m + n\log n + d(m+n))/T)$	Linear
Bipartite	$O((d(m+n) + (m+n)\log(m+n))/T)$	
Ranking	$\sim O(dmn + mn\log(mn)/\sqrt{T})$	Linear

Table 1: Complexity comparison with SOTA approaches

Convergence and Computational Complexity

Follows immediately from the convergence result of Nesterov's method, we have:

Theorem 3. *Let $\boldsymbol{\alpha}_T$ and $\boldsymbol{\beta}_T$ be the output from the τ-FPL algorithm after T iterations, then $g(\boldsymbol{\alpha}_T, \boldsymbol{\beta}_T) \leq \min g(\boldsymbol{\alpha}, \boldsymbol{\beta}) + \epsilon$, where $T \geq O(1/\sqrt{\epsilon})$.*

Finally, the computational cost of each iteration is dominated by the gradient evaluation and the projection step. Since the complexity of projection step is $O(m+n)$ and the cost of computing the gradient is $O(d(m+n))$, combining with Theorem 3 we have that: to find an ϵ-suboptimal solution, the total computational complexity of τ-FPL is $O(d(m+n)/\sqrt{\epsilon})$. Table 1 compares the computational complexity of τ-FPL with that of some state-of-the-art methods. The order of validation complexity corresponds to the number of hyper-parameters. From this, it is easy to see that τ-FPL is asymptotically more efficient.

Out-of-Bootstrap Thresholding

In the thresholding stage, the task is to identify the boundary between the positive instances and $(1-\tau)$ percent of the negative instances. Though thresholding on the training set is commonly used in (Joachims 1996; Davenport, Baraniuk, and Scott 2010; Scheirer et al. 2013), it may introduce overfitting. Hence, we propose an out-of-bootstrap method to find a more accurate and stable threshold. At each time, we randomly split the training set into two sets $\mathcal{S}_1$ and $\mathcal{S}_2$, and then train on $\mathcal{S}_1$ as well as the select threshold on $\mathcal{S}_2$. The procedure can be running multiple rounds to make use of all the training data. Once the process is completed, we can obtain the final threshold by averaging. On the other hand, the final scoring function can be obtained by two ways: learn a scoring function using the full set of training data, or gather the weights learned in each previous round and average them. This method combines both the advantages of out-of-bootstrap and soft-thresholding techniques: accurate error estimating and reduced variance with little sacrifice on bias, thus fits the setting of thresholding near the risk area.

Theoretical Guarantees

Now we develop the theoretical guarantee for the scoring function, which bounds the probability of giving any positive instances higher score than $1-\tau$ proportion of negative instances. To this end, we first define $h(\boldsymbol{x}, f)$, the probability for any negative instance to be ranked above x using f, i.e. $h(x, f) = \mathbb{E}_{x^- \sim \mathbb{P}^-}[\mathbb{I}(f(\boldsymbol{x}) \leq f(\boldsymbol{x}^-))]$, and then measure the quality of f by $P(f, \tau) = \mathbb{P}_{\boldsymbol{x}^+ \sim \mathbb{P}^+}(h(\boldsymbol{x}^+, f) \geq \tau)$, which is the probability of giving any positive instances lower score than τ percent of negative instances. The following theorem bounds $P(f, \tau)$ by the empirical loss $L_{\bar{k}}$.

Theorem 4. *Given training data $\mathcal{S}$ consisting of m independent instances from distribution $\mathbb{P}^+$ and n independent instances from distribution $\mathbb{P}^-$, let f^* be the optimal solution to the problem (6). Assume $m \geq 12$ and $n \gg s$. We have, for any $k \leq n$, with a probability at least $1 - 2e^{-s}$,*

$$P(f^*, \tau) \leq L_{\bar{k}} + O(\sqrt{(s + \log(m)/m)}) \tag{18}$$

where $\tau = O(\sqrt{\log m/n} + k/n)$, and $L_{\bar{k}} = \frac{1}{m}\sum_{i=1}^{m} l(f^*(\boldsymbol{x}_{\boldsymbol{i}}^+) - \frac{1}{k}\sum_{j=1}^{k} f^*(\boldsymbol{x}_{[\boldsymbol{j}]}^-))$.

Proof. Due to space limit, the proof is put in appendix. □

Theorem 4 implies that if $L_{\bar{k}}$ is upper bounded by $O(\log(m)/m))$, the probability of ranking any positive samples below τ percent of negative samples is also bounded by $O(\log(m)/m))$. If m is approaching infinity, $P(f^*, \tau)$ would be close to 0, which means in that case, we can almost ensure that by thresholding at a suitable point, the true-positive rate will get close to 1. Moreover, we observe that m and n play different roles in this bound. For instance, it is well known that the largest absolute value of Gaussian random instances grows in $\log(n)$. Thus we believe that the growth of n only slightly affects both the largest and the centroid of top-proportion scores of negatives samples. This leads to a conclusion that increasing n only slightly raise $L_{\bar{k}}$, but significant reduce the margin between target τ and k/n. On the other hand, increasing m will reduce upper bound of P, thus increasing the chance of finding positive instances at the top. In sum, n and m control τ and P respectively.

Experiment Results

Effectiveness of the Linear-time Projection

We first demonstrate the effectiveness of our projection algorithm. Following the settings of (Liu and Ye 2009), we randomly sample 1000 samples from the normal distribution $\mathcal{N}(0, 1)$ and solve the projection problem. The comparing method is *ibis* (Liu and Ye 2009), an improved bisection algorithm which also makes use of the convexity and monotonicity. All experiments are running on an Intel Core i5 Processor. As shown in Fig.1, thanks to the efficient reduction of the constant factor, our method outperforms *ibis* by saving almost 75% of the running time in the limit case.

We also solve the projection problem proposed in (Lapin, Hein, and Schiele 2015) by using a simplified version of our method, and compare it with the method presented in (Lapin, Hein, and Schiele 2015) (PTkC), whose complexity is $O(n\log(n) + kn)$. As one can observe from Fig.1(b), our method is linear in complexity regarding with n and does not suffer from the growth of k. In the limit case (both large k and n), it is even 3-order faster than the competitors.

Ranking Performance

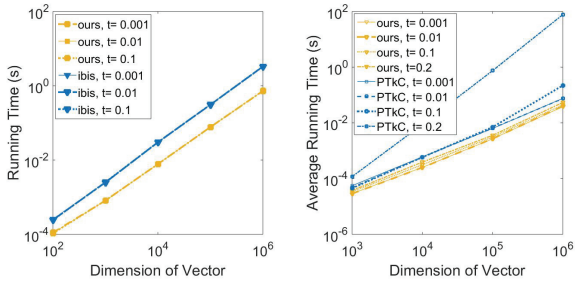
Next, we validate the ranking performance of our τ-FPL method, i.e. scoring and sorting test samples, and then evaluate the proportion of positive samples ranked above $1-\tau$

Dataset	heart 120/150,d:13		spambase 1813/2788,d:57					real-sim 22238/50071,d:20958					w8a 2933/62767,d:300					
$\tau(\%)$	5	10	0.1	0.5	1	5	10	0.01	0.1	1	5	10	0.05	0.1	0.5	1	5	10
CS-SVM	.526	.691	.109	.302	**.487**	.811	.920	.376	**.748**	.921	.972	.990	.501	.520	.649	.695	.828	.885
TopPush	**.541**	.711	.112	.303	.484	.774	.845	**.391**	.747	.920	.968	.983	**.508**	**.551**	.627	.656	.761	.842
$\text{SVM}^{\text{pAUC}}_{\text{tight}}$	.509	.728	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
τ-Rank	**.541**	**.740**	**.112**	**.305**	.460	**.842**	**.929**	**.391**	.747	.920	**.975**	**.991**	**.508**	**.551**	.645	**.710**	**.832**	**.894**
2τ-Rank	**.547**	**.734**	**.112**	**.311**	.477	**.862**	**.936**	**.391**	.747	**.922**	**.978**	**.992**	**.508**	.549	**.675**	**.739**	**.841**	**.902**

Table 2: Ranking performance by different values of the tolerance τ. The number of positive/negative instances and feature dimensions ('d') is shown together with the name of each dataset. The best results are shown in bold. 'N/A's denote the experiments that require more than one week for training.

Dataset(+/-)	$\tau(\%)$	BS-SVM	CS-LR	CS-SVM	CS-SVM-OOB	τ-FPL	2τ-FPL
heart	5	(.069, .675), .713	(.035, .394), .606	(.027, .327), .673	(.058, .553), .609	(.053, .582), **.468**	(.055, .584), **.514**
120/150,d:13	10	(.121, .774), .435	(.058, .615), .385	(.078, .666), .334	(.088, .682), .318	(.086, .686), **.314**	(.080, .679), **.317**
breast-cancer	1	(.015, .964), .559	(.007, .884), **.116**	(.006, .870), .130	(.014, .955), .451	(.013, .955), .324	(.011, .949), .192
239/444	5	(.063, .978), .276	(.013, .965), .035	(.017, .965), .034	(.046, .974), **.026**	(.041, .976), **.025**	(.045, .974), **.026**
d:10	10	(.113, .985), .142	(.035, .970), .030	(.044, .973), .027	(.095, .981), .020	(.098, .982), **.018**	(.094, .982), **.018**
spambase	0.5	(.008, .426), 1.220	(.007, .011), 1.362	(.002, .109), .891	(.005, .275), .790	(.005, .278), **.722**	(.004, .268), **.732**
1813/2788	1	(.013, .583), .748	(.007, .011), .989	(.004, .256), .744	(.009, .418), .582	(.008, .416), .584	(.008, .440), **.560**
d:57	5	(.054, .895), .192	(.007, .011), .989	(.020, .667), .333	(.047, .793), .207	(.041, .822), **.178**	(.046, .845), **.155**
	10	(.103, .941), .087	(.007, .011), .989	(.051, .716), .284	(.090, .902), .099	(.087, .925), **.075**	(.090, .928), **.072**
real-sim	0.01	(.002, .813), 22.376	(.001, .207), 7.939	(.000, .209), .791	(.000, .268), .833	(.000, .270), **.730**	(.000, .270), **.730**
22238/50071	0.1	(.008, .919), 7.09	(.001, .207), .826	(.001, .700), .428	(.001, .584), .416	(.001, .585), **.415**	(.001, .585), **.415**
d:20958	0.5	(.023, .966), 3.680	(.001, .207), .794	(.001, .755), .245	(.003, .810), .190	(.003, .829), **.174**	(.003, .827), **.181**
	1	(.036, .978), 2.570	(.001, .207), .794	(.007, .880), .121	(.007, .875), .125	(.007, .894), **.115**	(.006, .891), **.109**
	5	(.094, .994), .878	(.078, .994), .575	(.029, .931), .139	(.039, .965), .035	(.041, .972), **.028**	(.044, .974), **.028**
	10	(.133, 0.997), .336	(.078, .994), **.007**	(.069, .993), .007	(.099, .986), .019	(.092, .991), .009	(.094, .991), .009
w8a	0.05	(.001, .525), .966	(.000, .101), .900	(.000, .420), .580	(.000, .438), **.562**	(.000, .428), .572	(.000, .428), .572
1933/62767	0.1	(.001, .585), .710	(.000, .119), .881	(.000, .447), .553	(.001, .493), .507	(.001, .495), **.505**	(.001, .499), **.501**
d:123	0.5	(.006, .710), .437	(.000, .119), .881	(.002, .595), .405	(.003, .634), .366	(.003, .654), **.347**	(.003, .667), **.333**
	1	(.011, .749), .341	(.014, .696), .715	(.006, .642), .358	(.006, .695), .305	(.006, .702), **.298**	(.007, .726), **.274**
	5	(.048, .823), .177	(.014, .696), .305	(.013, .701), .299	(.046, .805), .195	(.033, .818), .182	(.036, .827), **.173**
	10	(.049, .823), .177	(.014, .696), .305	(.013, .701), .299	(.053, .814), .186	(.042, .833), **.167**	(.038, .826), **.174**

Table 3: [(mean false positive rate, mean true positive rate), NP-score] on real-world datasets by different values of the tolerance τ. In the leftmost column, the number of positive/negative instances and feature dimensions ('d') in each dataset. For each dataset, the best results are shown in bold.



(a) Run time against the method ibis (log-log).

(b) Run time against PTkC (log-log).

Figure 1: Running time against two peer projection methods.

proportion of negative samples. Considering ranking performance independently can avoid the practical problem of mismatching the constraint in (2) on testing set, and always offer us the optimal threshold.

Specifically, we choose (3) as evaluation and validation criterion. Compared methods include cost-sensitive SVM (CS-SVM) (Osuna, Freund, and Girosi 1997), which has been shown a lower bound approximation of (3); TopPush (Li, Jin, and Zhou 2014) ranking, which focus on optimizing the absolute top of the ranking list, also a special case of our model ($\tau = 0$); $\text{SVM}^{\text{pAUC}}_{\text{tight}}$ (Narasimhan and Agarwal 2013b), a more general method which designed for optimizing arbitrary partial-AUC. We test two version of our algorithms: τ-Rank and 2τ-Rank, which correspond to the different choice of τ in learning scheme. Intuitively, enlarge τ in training phase can be seen as a top-down approximation—from upper bound to the original objective (2). On the other hand, the reason for choosing 2τ is that, roughly speaking, the average score of the top 2τ proportion of negative samples may close to the score of $\lceil n\tau \rceil$-th negative sample.

Settings. We evaluate the performance on publicly benchmark datasets with different domains and various sizes[3]. For small scale datasets($\leq 10,000$ instances), 30 times stratified hold-out tests are carried out, with $2/3$ data as train set and $1/3$ data as test set. For large datasets, we instead run 10 rounds. In each round, hyper-parameters are chosen by 5-fold cross validation from grid, and the search scope is extended if the optimal is at the boundary.

Results. Table 2 reports the experimental results. We note that at most cases, our proposed method outperforms other

[3]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary

peer methods. It confirms the theoretical analysis that our methods can extract the capacity of the model better. For TopPush, it is highly-competitive in the case of extremely small τ, but gradually lose its advantage as τ increase. The algorithm of $\text{SVM}^{\text{pAUC}}_{\text{tight}}$ is based on cutting-plane methods with exponential number of constraints, similar technologies are also used in many other ranking or structured prediction methods, e.g. Structured SVM (Tsochantaridis et al. 2005). The time complexity of this kind of methods is $O((m+n)\log(m+n))$, and we found that even for thousands of training samples, it is hard to finish experiments in allowed time.

Overall Classification Accuracy

In this section we compare the performance of different models by jointly learning the scoring function and threshold in training phase, i.e. output a classifier. To evaluate a classifier under the maximum tolerance, we use *Neyman-Pearson score* (NP-score) (Scott 2007). The NP-score is defined by $\frac{1}{\tau}\max\{fpr, \tau\} - tpr$ where fpr and tpr are false-positive rate and true-positive rate of the classifier respectively, and τ is the maximum tolerance. This measure punishes classifiers whose false-positive rates exceed τ, and the punishment becomes higher as $\tau \to 0$.

Settings. We use the similar setting for classification as for ranking experiments, i.e., for small scale datasets, 30 times stratified hold-out tests are carried out; for large datasets, we instead run 10 rounds. Comparison baselines include: Cost-Sensitive Logistic Regression (CS-LR) which choose a surrogate function that different from CS-SVM; Bias-Shifting Support Vector Machine (BS-SVM), which first training a standard SVM and then tuning threshold to meet specified false-positive rate; cost-sensitive SVM (CS-SVM). For complete comparison, we also construct a CS-SVM by our out-of-bootstrap thresholding (CS-SVM-OOB), to eliminate possible performance gains comes from different thresholding method, and focus on the training algorithm itself. For all of comparing methods, the hyper-parameters are selected by 5-fold cross-validation with grid search, aims at minimizing the NP-score, and the search scope is extended when the optimal value is at the boundary. For our τ-FPL, in the ranking stage the regularization parameter R is selected to minimize (3) , and then the threshold is chosen to minimize NP-score. We test two variants of our algorithms: τ-FPL and 2τ-FPL, which corresponding different choice of τ in learning scheme. As mentioned previously, enlarge τ can be seen as a top-down approximation towards the original objective.

Results. The NP-score results are given in Table 3. First, we note that both our methods can achieve the best performance in most of the tests, compared to various comparing methods. Moreover, it is clear that even using the same method to select the threshold, the performance of cost sensitive method is still limited. Another observation is that both of the three algorithms which using out-of-bootstrap thresholding can efficiently control the false positive rate under the constraint. Moreover, τ-FPLs are more stable than other algorithms, which we believe benefits from the accurate splitting of the positive-negative instances and stable thresholding techniques.

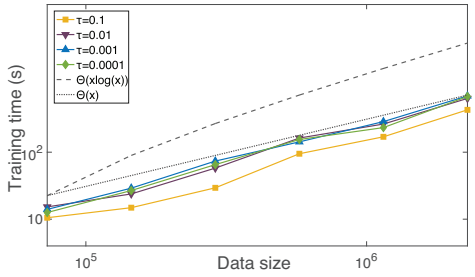


Figure 2: Training time of τ-FPL versus training data size for different τ (log-log).

Scalability

We study how τ-FPL scales to a different number of training examples by using the largest dataset real-sim. In order to simulate the limit situation, we construct six datasets with different data size, by up-sampling original dataset. The sampling ratio is $\{1, 2, 2^2, ...2^5\}$, thus results in six datasets with data size from 72309 to 2313888. We running τ-FPL ranking algorithm on these datasets with different τ and optimal R (chosen by cross-validation), and report corresponding training time. Up-sampling technology ensures that, for a fixed τ, all the six datasets share the same optimal regularization parameter R. Thus the unique variable can be fixed as data size. Figure 2 shows the log-log plot for the training time of τ-FPL versus the size of training data, where different lines correspond to different τ. It is clear that the training time of τ-FPL is indeed linear dependent in the number of training data. This is consistent with our theoretical analysis and also demonstrate the scalability of τ-FPL.

Conclusion

In this paper, we focus on learning binary classifier under the specified tolerance τ. To this end, we have proposed a novel ranking method which directly optimizes the probability of ranking positive samples above $1-\tau$ percent of negative samples. The ranking optimization is then efficiently solved using projected gradient method with the proposed linear time projection. Moreover, an out-of-bootstrap thresholding is applied to transform the learned ranking model into a classifier with a low false-positive rate. We demonstrate the superiority of our method using both theoretical analysis and extensive experiments on several benchmark datasets.

Acknowledgments

The mainly work was done when the first author was an intern at iDST of Alibaba. This work is supported by the National Natural Science Foundation of China (NSFC) (61702186, 61672236, 61602176, 61672231), the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Information (U1609220), the Key Program of Shanghai Science and Technology Commission (15JC1401700) and the Joint Research Grant Proposal for Overseas Chinese Scholars (61628203).

References

Ben-Hur, A.; Horn, D.; Siegelmann, H. T.; and Vapnik, V. 2001. Support vector clustering. *JMLR* 2(Dec):125–137.

Davenport, M. A.; Baraniuk, R. G.; and Scott, C. D. 2006. Controlling false alarms with support vector machines. In *ICASSP*, volume 5, V–V.

Davenport, M. A.; Baraniuk, R. G.; and Scott, C. D. 2010. Tuning support vector machines for minimax and neyman-pearson classification. *TPAMI* 32(10):1888–1898.

Drucker, H.; Wu, D.; and Vapnik, V. N. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks* 10(5):1048–1054.

Gasso, G.; Pappaioannou, A.; Spivak, M.; and Bottou, L. 2011. Batch and online learning algorithms for nonconvex neyman-pearson classification. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):28.

Huang, H.; Liu, C.-C.; and Zhou, X. J. 2010. Bayesian approach to transforming public gene expression repositories into disease diagnosis databases. *PNAS* 107(15):6823–6828.

Joachims, T. 1996. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document.

Kiwiel, K. C. 2005. On floyd and rivest's select algorithm. *Theoretical Computer Science* 347(1-2):214–238.

Kiwiel, K. C. 2007. On linear-time algorithms for the continuous quadratic knapsack problem. *Journal of Optimization Theory and Applications* 134(3):549–554.

Lapin, M.; Hein, M.; and Schiele, B. 2015. Top-k multiclass SVM. In *NIPS*, 325–333.

Lehmann, E. L., and Romano, J. P. 2006. *Testing statistical hypotheses*. Springer Science & Business Media.

Li, N.; Jin, R.; and Zhou, Z.-H. 2014. Top rank optimization in linear time. In *NIPS*, 1502–1510.

Liu, J., and Ye, J. 2009. Efficient euclidean projections in linear time. In *ICML*, 657–664.

Liu, X.-Y., and Zhou, Z.-H. 2010. Learning with cost intervals. In *KDD*, 403–412.

Mahdavi, M.; Yang, T.; and Jin, R. 2013. Stochastic convex optimization with multiple objectives. In *NIPS*. 1115–1123.

Masnadi-Shirazi, H., and Vasconcelos, N. 2007. Asymmetric boosting. In *ICML*, 609–619.

Masnadi-Shirazi, H., and Vasconcelos, N. 2011. Cost-sensitive boosting. *TPAMI* 33(2):294–309.

Mozer, M. C.; Dodier, R.; Colagrosso, M. D.; Guerra-Salcedo, C.; and Wolniewicz, R. 2002. Prodding the roc curve: Constrained optimization of classifier performance. In *NIPS*, 1409–1415.

Narasimhan, H., and Agarwal, S. 2013a. On the relationship between binary classification, bipartite ranking, and binary class probability estimation. In *NIPS*, 2913–2921.

Narasimhan, H., and Agarwal, S. 2013b. A structural svm based approach for optimizing partial auc. In *ICML*.

Nesterov, Y. 2003. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers.

Osuna, E.; Freund, R.; and Girosi, F. 1997. Support vector machines: Training and applications. Technical Report AIM-1602.

Patriksson, M. 2008. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research* 185(1):1–46.

Rigollet, P., and Tong, X. 2011. Neyman-pearson classification, convexity and stochastic constraints. *JMLR* 12(Oct):2831–2855.

Scheirer, W. J.; de Rezende Rocha, A.; Sapkota, A.; and Boult, T. E. 2013. Toward open set recognition. *TPAMI* 35(7):1757–1772.

Scott, C., and Nowak, R. 2005. A neyman-pearson approach to statistical learning. *IEEE Transactions on Information Theory (TIT)* 51(11):3806–3819.

Scott, C. 2007. Performance measures for neyman–pearson classification. *IEEE Transactions on Information Theory (TIT)* 53(8):2852–2863.

Tong, X. 2013. A plug-in approach to neyman-pearson classification. *JMLR* 14(Oct):3011–3040.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *JMLR* 6(Sep):1453–1484.

Wu, S.-H.; Lin, K.-P.; Chen, C.-M.; and Chen, M.-S. 2008. Asymmetric support vector machines: low false-positive learning under the user tolerance. In *KDD*, 749–757.

Zhou, Y.-H., and Zhou, Z.-H. 2016. Large margin distribution learning with cost interval and unlabeled data. *TKDE* 28(7):1749–1763.