Optimizing Non-decomposable Performance Measures: A Tale of Two Classes

Harikrishna Narasimhan[*] HARIKRISHNA@CSA.IISC.ERNET.IN
Indian Institute of Science, Bangalore, INDIA

Purushottam Kar T-PURKAR@MICROSOFT.COM
Microsoft Research, INDIA

Prateek Jain PRAJAIN@MICROSOFT.COM
Microsoft Research, INDIA

Abstract

Modern classification problems frequently present mild to severe label imbalance as well as specific requirements on classification characteristics, and require optimizing performance measures that are non-decomposable over the dataset, such as F-measure. Such measures have spurred much interest and pose specific challenges to learning algorithms since their non-additive nature precludes a direct application of well-studied large scale optimization methods such as stochastic gradient descent.

In this paper we reveal that for two large families of performance measures that can be expressed as functions of true positive/negative rates, it is indeed possible to implement point stochastic updates. The families we consider are concave and pseudo-linear functions of TPR, TNR which cover several popularly used performance measures such as F-measure, G-mean and H-mean.

Our core contribution is an adaptive linearization scheme for these families, using which we develop optimization techniques that enable truly point-based stochastic updates. For concave performance measures we propose **SPADE**, a stochastic primal dual solver; for pseudo-linear measures we propose **STAMP**, a stochastic alternate maximization procedure. Both methods have crisp convergence guarantees, demonstrate significant speedups over existing methods - often by an order of magnitude or more, and give similar or more accurate predictions on test data.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[*]Part of the work was done while H.N. was an intern at MSR India.

1. Introduction

Learning applications with binary classification problems involving severe label imbalance abound, often accompanied with specific requirements in terms of false positive, or negative rates. Examples included spam classification, anomaly detection, and medical applications. Class imbalance is also often introduced as a result of the reduction of a problem to binary classification, such as in multi-class problems (Bishop, 2006) and multi-label problems due to extreme label sparsity (Hsu et al., 2009).

Traditional performance measures such as misclassification rate are ill-suited in such situations as it is usually trivial to optimize them by constantly predicting the majority class. Instead, the performance measures of choice in such cases are those that perform a more holistic evaluation over the entire data. Naturally, these performance measures are *non-decomposable* over the dataset and cannot be cannot be expressed as a sum of errors on individual data points. Popular examples include F-measure, G-mean, H-mean etc.

A consistent effort directed at optimizing these performance measures has, over the years, resulted in the development of two broad approaches - 1) surrogate based approaches (e.g. SVMPerf (Joachims et al., 2009)) that design convex surrogates for these performance measures, and 2) indirect approaches which include cost-sensitive classification-based approaches (Parambath et al., 2014) which solve weighted classification problems, and plug-in approaches (Koyejo et al., 2014; Narasimhan et al., 2014) which rely on consistent estimates of class probabilities.

Both these approaches are known to work fairly well on small datasets but do not scale very well to large ones, especially those large enough to not even fit in memory. SVMPerf-style approaches, which employ cutting plane methods do not scale well. On the other hand, plug-in approaches first need to solve a class probability estimation problem optimally and then tune a threshold. This two-

stage approach prevents the method from exploiting better classifiers to automatically obtain better thresholds. Moreover, for multi-class problems with C classes, jointly estimating C parameters can take time exponential in C.

For large datasets, streaming methods such as stochastic gradient descent (Shalev-Shwartz et al., 2011) that take only a few passes over the entire data are preferable. However, traditional SGD techniques cannot handle non-decomposable losses. Recently, (Kar et al., 2014) proposed optimizing SVMPerf-style surrogates using SGD techniques. Although their method is generic, allowing optimization of performance measures such as F-measure and partial AUC, they require maintaining a *large buffer* to compute online gradient estimates that can be prohibitive.

Motivated by the state of the art, we develop novel methods for optimizing two broad families of non-decomposable performance measures. Our methods incorporate truly point-wise updates, i.e. do not require a buffer, and require only a few passes over data. At an intuitive level, at the core of our work are adaptive linearization strategies for these performance measures, which make these measures amenable to SGD-style point-wise updates. Moreover, our linearizations are able to feed off the improvements made in learning a better classifier, resulting in faster convergence.

We consider two classes of performance measures

Concave Performance Measures (see Table 1): These measures can be written as concave functions of true positive (TPR) and negative (TNR) rates and include G-mean, H-mean etc. We exploit the dual structure of these functions via their Fenchel dual to linearize them in terms of the TPR, TNR variables. Our method then, in parallel, tunes the dual variables in this linearization and maximizes the weighted TPR-TNR combination. These updates are done in an online fashion using stochastic mirror descent steps.

Pseudo-linear Performance Measures (see Table 2): These measures can be written as fractional linear functions of TPR, TNR and include F-measure and the Jaccard coefficient. These functions need not be concave and the techniques outlined above do not apply. Instead, we exploit the pseudo-linear structure to linearize the function and develop a technique to alternately optimize the combination weights and the classifier model via stochastic updates. Although such "alternate-maximization" strategies in general need not converge even to a local optima, we show that our strategy converges to an ϵ-approximate global optimum after $\log\left(\frac{1}{\epsilon}\right)$ batch updates or $O(1/\epsilon^2)$ stochastic updates.

Finally, we present an empirical validation of our methods. Our experiments reveal that for a range of performance measures in both classes, our methods can be significantly faster than either plug-in or SVMPerf-style methods, as well as give higher or comparable accuracies.

2. Related Works

As noted in Section 1, existing methods for optimizing performance measures that we study can be divided into surrogate-based approaches and indirect approaches based on cost-sensitive classification or plug-in methods. A third approach applicable to certain performance measures is the *decision-theoretic* method that learns a class probability estimate and computes predictions that maximize the expected value of the performance measure on a test set (Lewis, 1995; Ye et al., 2012). In addition to these there exist methods dedicated to specific performance measures.

For instance (Parambath et al., 2014) focus on optimizing F-measure by exploiting the pseudo-linearity of the function along with a cross validation-based strategy. Our **STAMP** method, on the other hand uses an alternating maximization strategy that does not require cross validation which considerably improves training time (see Figure 3). It is important to note that these performance measures have also been studied in multi-label settings where these no longer remain non-decomposable. For instance, (Dembczyński et al., 2013) study plug-in style methods for maximizing F-measure in multi-label settings whereas works such as (Koyejo et al., 2014; Narasimhan et al., 2014; Ye et al., 2012) study plug-in approaches for the same problem in the more challenging binary classification setting.

Historically, online learning algorithms have played a key role in designing solvers for large-scale batch problems. However, for non-decomposable loss functions, defining an online learning framework and providing efficient algorithms with small regret itself is challenging. (Rakhlin et al., 2011) propose a generic method for such loss functions; however the algorithms proposed therein run in exponential time. (Kar et al., 2014) also study such measures with the aim of designing stochastic gradient-style methods. However, their methods require a large buffer to be maintained, which causes them to have poorer convergence guarantees and in practice be slower than our methods.

By exploiting the special structure in our function classes, we are able to do away with such requirements. Our methods make use of standard online convex optimization primitives (Zinkevich, 2003). However, their application requires special care in order to avoid divergent behavior.

3. Problem Setting

Let $\mathcal{X} \subset \mathbb{R}^d$ denote the instance space and $\mathcal{Y} = \{-1, +1\}$ the label space, with some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Let $p := \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[y = +1]$ denote the proportion of positives in the population. Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$ denote a sample of training points sampled i.i.d. from $\mathcal{D}$. For sake of simplicity we shall present our algorithms and analyses for

a set of linear models $\mathcal{W} \subseteq \mathbb{R}^d$. Let $R_{\mathcal{X}}$ and $R_{\mathcal{W}}$ denote the radii of the domain $\mathcal{X}$ and hypothesis class $\mathcal{W}$ respectively.

We consider performance measures that can be expressed in terms of the true positive and negative rates of a classifier. To represent these measures, we shall use the notion of a *reward function* r that assigns a *reward* $r(y, \hat{y})$ to a prediction $\hat{y} \in \mathbb{R}$ when the true label is $y \in \mathcal{Y}$. We will use

$$r^+(\mathbf{w}; \mathbf{x}, y) = \frac{1}{p} \cdot r(y, \mathbf{w}^\top \mathbf{x}) \cdot \mathbf{1}(y = 1)$$

$$r^-(\mathbf{w}; \mathbf{x}, y) = \frac{1}{1-p} \cdot r(y, \mathbf{w}^\top \mathbf{x}) \cdot \mathbf{1}(y = -1)$$

to calculate rewards on positive and negative points. Since $\mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r^+(\mathbf{w}; \mathbf{x}, y)]\!] = \mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r(y, \mathbf{w}^\top \mathbf{x}) | y = 1]\!]$, setting $r^{0-1}(y, \hat{y}) = \mathbf{1}(y\hat{y} > 0)$ gives us $\mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r^+(\mathbf{w}; \mathbf{x}, y)]\!] = \text{TPR}(\mathbf{w})$. For sake of convenience, we will use $P(\mathbf{w}) = \mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r^+(\mathbf{w}; \mathbf{x}, y)]\!]$ and $N(\mathbf{w}) = \mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r^-(\mathbf{w}; \mathbf{x}, y)]\!]$ to denote population averages of the reward functions. We shall assume that our reward functions are concave, L_r-Lipschitz, and take values in a bounded range $[-B_r, B_r]$.

4. Concave Performance Measures

The first class of performance measures we analyze are concave performance measures. These measures can be written as concave functions of the TPR and TNR i.e.

$$\mathcal{P}_\Psi(\mathbf{w}) = \Psi\left(P(\mathbf{w}), N(\mathbf{w})\right)$$

for some concave link function $\Psi : \mathbb{R}^2 \to \mathbb{R}$. A large number of popular performance measures fall in this family since these measures are relevant in situations with severe label imbalance or in situations where cost-sensitive classification is required such as detection theory (Vincent, 1994). Table 1 gives a list of such performance measures along with some of their relevant properties and references to works that utilize these performance measures.

We shall find it convenient to define the (concave) Fenchel conjugate of the link functions for our performance measures. For any concave function Ψ and $\alpha, \beta \in \mathbb{R}$, define

$$\Psi^*(\alpha, \beta) = \inf_{u,v \in \mathbb{R}} \left\{\alpha u + \beta v - \Psi(u, v)\right\}.$$

By the concavity of Ψ, we have, for any $u, v \in \mathbb{R}$,

$$\Psi(u, v) = \inf_{\alpha,\beta \in \mathbb{R}} \left\{\alpha u + \beta v - \Psi^*(\alpha, \beta)\right\}.$$

We shall use the notation Ψ to denote, both the link function, as well as the performance measure it induces.

4.1. A Stochastic Primal-dual Method for Optimizing Concave Performance Measures

We now present a novel online stochastic method for optimizing the class of concave performance measures. The use of stochastic gradient techniques for these measures presents specific challenges due to the non-decomposable nature of these measures which makes it difficult to obtain cheap, unbiased estimates of the gradient using a single point. Recent works (Kar et al., 2013; 2014) have tried to resolve this issue by looking at mini-batch methods or by using a buffer to maintain a sketch of the stream. However, such techniques bring in a bias into the learning algorithm in the form of buffer size or mini batch length which results in slower convergence. Indeed, the **1PMB** method of (Kar et al., 2014) is only able to guarantee a $\sqrt[-4]{T}$ rate of convergence, whereas SGD techniques are usually able to guarantee $\sqrt[-2]{T}$ rates. This is indicative of suboptimal performance and our experiments confirm this (see Figure 3).

Algorithm 1 **SPADE**: Stochastic PrimAl-Dual mEthod

Input: Primal/dual step sizes η_t, η'_t, feasible sets $\mathcal{W}, \mathcal{A}$
Output: Classifier $\mathbf{w} \in \mathcal{W}$

$\mathbf{w}_0 \leftarrow \mathbf{0}, t \leftarrow 1$
while data stream has points **do**
  Receive data point $(\mathbf{x}_t, y_t)$
  /* *Perform primal ascent* */
  if $y_t > 0$ **then**
    $\mathbf{w}_{t+1} \leftarrow \Pi_{\mathcal{W}} \left(\mathbf{w}_t + \eta_t \cdot \alpha_t \nabla_{\mathbf{w}} r^+(\mathbf{w}_t; \mathbf{x}_t, y_t)\right)$
  else
    $\mathbf{w}_{t+1} \leftarrow \Pi_{\mathcal{W}} \left(\mathbf{w}_t + \eta_t \cdot \beta_t \nabla_{\mathbf{w}} r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)\right)$
  end if
  /* *Perform dual descent* */
  $(a, b) \leftarrow (\alpha_t, \beta_t) - \eta'_t \cdot \nabla_{(\alpha,\beta)} \Psi^*(\alpha_t, \beta_t)$
  if $y_t > 0$ **then**
    $a \leftarrow a - \eta'_t \cdot r^+(\mathbf{w}_t; \mathbf{x}_t, y_t)$
  else
    $b \leftarrow b - \eta'_t \cdot r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)$
  end if
  $(\alpha_{t+1}, \beta_{t+1}) \leftarrow \Pi_{\mathcal{A}}((a, b))$
  $t \leftarrow t + 1$
end while
return $\overline{\mathbf{w}} = \frac{1}{t} \sum_{\tau=1}^{t} \mathbf{w}_\tau$

Here we show that for the class of concave performance measures, such workarounds are not necessary. To this end we present the **SPADE** algorithm (Algorithm 1) which exploits the dual structure of the performance measures to obtain efficient point updates which do not require the use of mini-batches or online buffers. **SPADE** is able to offer convergence guarantees identical to those that stochastic methods offer for additive performance measures such as least squares, without the presence of any algorithmic bias.

Let $\mathcal{W} \subset \mathcal{X}$ and $\mathcal{A}_\Psi \subset \mathbb{R}^2$ be convex regions within the model and dual spaces respectively, and $\Pi_{\mathcal{W}}$ and $\Pi_{\mathcal{A}_\Psi}$ denote projection operators for these. Table 1 lists the relevant dual regions for the performance measures listed therein.

4.2. Convergence Analysis for SPADE

This section presents a convergence analysis for the **SPADE** algorithm. The convergence proof is formally

Table 1. List of concave performance measures $\Psi(P, N)$ along with their monotonicity and Lipschitz properties, sufficient dual regions, and expressions for dual subgradients. $\mathcal{B}(\mathbf{0}, r)$ denotes the ball of radius r around the origin. $\mathbb{R}^2_+$ denotes the positive quadrant.

Name	Expression (P, N)	Mon.?	Lip.?	$\delta(\epsilon)$	Sufficient dual Region $\mathcal{A}_\Psi$	$\nabla\Psi^*(\alpha, \beta)$.
Min ((Vincent, 1994))	$\min\{P, N\}$	✓	✓	ϵ	$\{\alpha + \beta = 1\} \cap \mathbb{R}^2_+$	$\mathbf{0}$
H-mean ((Kennedy et al., 2010))	$\frac{2PN}{P+N}$	✓	✓	4ϵ	$\left\{\sqrt{\alpha} + \sqrt{\beta} \geq \sqrt{2}\right\} \cap \mathcal{B}(\mathbf{0}, 2)$	$\mathbf{0}$
Q-mean ((Liu & Chawla, 2011))	$1 - \sqrt{\frac{(1-P)^2 + (1-N)^2}{2}}$	✓	✓	ϵ	$\left\{\alpha^2 + \beta^2 \leq 1/2\right\} \cap \mathbb{R}^2_+$	$\mathbf{1}$
G-mean ((Daskalaki et al., 2006))	$\sqrt{PN}$	✓	✗	$3\sqrt{\epsilon}$	$\{\alpha\beta \geq 1/4\} \cap \mathbb{R}^2_+$	$\mathbf{0}$

stated in Theorem 4. Apart from demonstrating the utility of the algorithm, the proof also sheds light on the choice of algorithm parameters, such as primal/dual feasible regions.

We shall work with performance measures that are monotonically increasing in the true positive and negative rates of the classifier i.e. if $u \geq u'$, $v \geq v'$ then $\Psi(u, v) \geq \Psi(u', v')$. This is a natural assumption and is satisfied by all performance measures considered here (see Table 1). We now introduce two useful concepts.

Definition 1 (Stable Performance Measure)**.** *A performance measure Ψ will be called* δ-stable *if for some function $\delta : \mathbb{R} \to \mathbb{R}$, we have for all $u, v \in \mathbb{R}$ and $\epsilon \in \mathbb{R}_+$,*

$$\Psi\left(u + \epsilon, v + \epsilon\right) \leq \Psi(u, v) + \delta(\epsilon).$$

Table 1 lists the stability parameters of all the concave performance measures. Clearly, a performance measure has a linear stability parameter i.e. $\delta(\epsilon) \leq L \cdot \epsilon$ iff its corresponding link function is Lipschitz. We now define the notion of a *sufficient dual region* for a performance measure

Definition 2 (Sufficient Dual Region)**.** *For any link function Ψ, define its* sufficient dual region $\mathcal{A}_\Psi \subseteq \mathbb{R}^2$ *to be the minimal set such that for all $(u, v) \in R^2$, we have*

$$\Psi(u, v) = \inf_{(\alpha,\beta)\in\mathcal{A}_\Psi} \left\{\alpha u + \beta v - \Psi^*(\alpha, \beta)\right\}.$$

The reason for defining this quantity will become clear in a moment. A closer look at Algorithm 1 indicates that it is performing online gradient descent steps with the dual variables. Clearly, for this procedure to have statistical convergence properties, the magnitude of the updates must be bounded in some sense otherwise the learning procedure may diverge. This motivates the projection step in Step 17. However, in order for the updated dual variables to be informative about the current primal function value, the projection step must be done in a way that does not distort the link function. The notion of a sufficient dual region formally captures the notion of such a projection step.

Having said that, there is no apriori guarantee that the sufficient region for a given performance measure would be bounded, in which case this entire exercise counts for naught. However, the following lemma, by closely linking the stability properties of a performance measure with the size of its sufficient dual region, shows that for well-behaved link functions, this will not be the case .

Lemma 3. *The stability parameter of a performance measure $\Psi(\cdot)$ can be written as $\delta(\epsilon) \leq L_\Psi \cdot \epsilon$ iff its sufficient dual region is bounded in a ball of radius L_Ψ.*

The proof of this result follows from elementary manipulations and can be found in Appendix A. In some sense this result can be seen as a realization of the well known connection between the Fenchel dual of a function and its Lipschitz properties.

To simplify the initial analysis, we shall first concentrate on performance measures whose link functions are Lipschitz. It is easy to see that these are exactly the performance measures whose gradients do not diverge within any compact region of the real plane. Of the performance measures listed in Table 1, all measures except G-mean have associated link functions that are Lipschitz. Subsequently, we shall address the more involved case of non-Lipschitz performance measures such as G-mean as well.

Theorem 4. *Suppose we are given a stream of random samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ drawn from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Let $\Psi(\cdot)$ be a concave, Lipschitz link function. Let Algorithm 1 be executed with a dual feasible set $\mathcal{A} \supseteq \mathcal{A}_\Psi$, $\eta_t = 1/\sqrt{t}$ and $\eta'_t = 1/\sqrt{t}$. Then, the average model $\overline{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$ output by the algorithm satisfies, with probability at least $1 - \delta$,*

$$\mathcal{P}_\Psi(\overline{\mathbf{w}}) \geq \sup_{\mathbf{w}^*\in\mathcal{W}} \mathcal{P}_\Psi(\mathbf{w}^*) - \mathcal{O}\left(\delta_\Psi\left(\sqrt{\frac{1}{T}\log\frac{1}{\delta}}\right)\right).$$

We refer the reader to Appendix B for a proof and explicit constants. The proof closely analyzes the primal ascent and dual descent steps, tying them together using the Fenchel dual of Ψ.

4.3. The Case of non-Lipschitz Link Functions

Non-Lipschitz link functions, such as the one used in the G-mean performance measure, pose a particular challenge to the previous analysis. Owing to their non-Lipschitz nature, their sufficient dual region is unbounded. Indeed as Table 1 indicates, the sufficient region for $\Psi_{\text{G-mean}}$ extends indefinitely along both coordinate axes. More precisely,

what happens is that the gradients of the $\Psi_{\text{G-mean}}$ function diverge as either $u \to 0$, or $v \to 0$. This poses a stumbling block for the proof of Theorem 4 since the regret and online-to-batch conversion results used therein fail.

A natural way to solve this problem is to ensure that the reward functions r^+, r^- always assign rewards that are bounded away from zero. More specifically, for some $\epsilon > 0$, we have $r^+(\mathbf{w}; \mathbf{x}, y), r^-(\mathbf{w}; \mathbf{x}, y) \geq \epsilon$ for all $\mathbf{w} \in \mathcal{W}$ and $\mathbf{x} \in \mathcal{X}$. For this restricted reward region, one can show, using Lemma 3, that the sufficient dual region can be restricted to a ball of radius $\mathcal{O}\left(\sqrt{1/\epsilon}\right)$.

The above discussion suggests that we regularize the reward function i.e. at each time step t, we add a small value $\epsilon(t)$ to the original reward function. However, the amount of regularization remains to be decided since over regularization could cause our resulting excess risk bound to be vacuous with respect to the original reward function. It turns out that setting $\epsilon(t) \approx \frac{1}{t^{1/4}}$ strikes a fine balance between regularization and fidelity to the original reward function - this seems intuitive since the regularization becomes milder and milder as learning progresses. The following extension of Theorem 4 formalizes this statement.

Theorem 5. *Suppose we have the problem setting in Theorem 4 with the $\Psi_{\text{G-mean}}$ performance measure being optimized for. Consider a modification to Algorithm 1 wherein the reward functions are changed to $r_t^+(\cdot) = r^+(\cdot) + \epsilon(t)$, and $r_t^-(\cdot) = r^-(\cdot) + \epsilon(t)$ for $\epsilon(t) = \frac{1}{t^{1/4}}$. Then, the average model $\overline{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$ output by the algorithm satisfies, with probability at least $1 - \delta$,*

$$\mathcal{P}_{\Psi_{\text{G-mean}}}(\overline{\mathbf{w}}) \geq \sup_{\mathbf{w}^* \in \mathcal{W}} \mathcal{P}_{\Psi_{\text{G-mean}}}(\mathbf{w}^*) - \widetilde{\mathcal{O}}\left(\frac{1}{T^{1/4}}\right).$$

The proof of this theorem can be found in Appendix C. We note here that primal dual frameworks have been utilized before in diverse areas such as distributed optimization (Jaggi et al., 2014) and multi-objective optimization (Mahdavi et al., 2013). However, these works simply assume the functions involved therein to be Lipschitz and/or smooth and do not address cases where they fail to be so. Theorem 5 on the other hand, is able to recover a non-trivial, albeit weaker, statement even for *locally Lipschitz* functions.

5. Pseudo-linear Performance Measures

The second class of performance measures we analyze are pseudo-linear performance measures. These measures have a fractional linear function as the link function and can be written as follows:

$$\mathcal{P}_{(\mathbf{a},\mathbf{b})}(\mathbf{w}) = \frac{a_0 + a_1 \cdot P(\mathbf{w}) + a_2 \cdot N(\mathbf{w})}{b_0 + b_1 \cdot P(\mathbf{w}) + b_2 \cdot N(\mathbf{w})},$$

for some weighing coefficients $\mathbf{a}, \mathbf{b}$. Several popularly used performance measures, most notably the F-measure, can be represented as pseudo-linear functions. Table 2 enumerates some popular pseudo-linear performance measures as well as their properties.

We note that these performance measures are usually represented in literature using the entries of the confusion matrix. However, for the sake of our analysis, we shall find it useful to represent them in terms of the true positive and true negative rates. To do so, we shall use p to denote the proportion of positives in the population and $\theta = \frac{1-p}{p}$ to denote the label skew.

5.1. Alternate-maximization for Optimizing Pseudo-linear Performance Measures

Pseudo-linear functions are named so since their level sets can be defined using linear half-spaces. More specifically, every pseudo-linear function Ψ over $\mathbb{R}^d$ has an associated "level-finder" function $a : \mathbb{R} \to \mathbb{R}^d$ and $b : \mathbb{R} \to \mathbb{R}$ such that $\Psi(\mathbf{v}) \geq t$ iff $\langle \mathbf{v}, a(t) \rangle \geq b(t)$. We refer the reader to (Parambath et al., 2014) for a more relaxed introduction to these functions and their properties. For our purposes, however, it suffices to notice that this property immediately points toward a cost-sensitive method to optimize these performance measures.

This fact was noticed by (Parambath et al., 2014) who exploited this to develop a cost-sensitive classification method for optimizing the F-measure by simply searching for the best weights with which to perform cost-sensitive classification. However, we notice that instead of performing such a brute force search, one can adaptively tune the weights to better and better values and obtain much faster convergence. To develop this intuition, we first define the notion of a *valuation function* below.

Definition 6 (Valuation Function)**.** *The* valuation function *of a performance measure $\mathcal{P}_{(\mathbf{a},\mathbf{b})}$, for a classifier $\mathbf{w} \in \mathcal{W}$, and at a level $v \in \mathbb{R}$ is defined as*

$$V_{(\mathbf{a},\mathbf{b})}(\mathbf{w}, v) := c + (\alpha - v\gamma) \cdot P(\mathbf{w}) + (\beta - v\delta) \cdot N(\mathbf{w}),$$

where $c = \frac{a_0}{b_0}, \alpha = \frac{a_1}{b_0}, \beta = \frac{a_2}{b_0}, \gamma = \frac{b_1}{b_0}, \delta = \frac{b_2}{b_0}$.

The following well-known lemma closely links the valuation function to the performance measure.

Lemma 7. *For any performance measure $\mathcal{P}_{(\mathbf{a},\mathbf{b})}$, $\mathbf{w} \in \mathcal{W}$ and $v \in \mathbb{R}$ we have $\mathcal{P}_{(\mathbf{a},\mathbf{b})}(\mathbf{w}) \geq v$ iff $V_{(\mathbf{a},\mathbf{b})}(\mathbf{w}, v) \geq v$. Moreover, in such a situation we say that classifier $\mathbf{w}$ has achieved valuation at* level v.

Lemma 7 indicates that the performance of a classifier is intimately linked to its valuation. This suggests a natural alternate maximization approach wherein we alternate between posing a *challenge level* to the classifier and training

Table 2. List of pseudo-linear performance measures $\mathcal{P}_{(\mathbf{a},\mathbf{b})}(P,N)$ along with their popular forms, canonical expressions in terms of (reward functions representative of) true positive (P) and negative (N) rates, monotonicity properties, acceptable range of reward values, and rate of convergence of the Alt-Max procedure for the performance measure when rewards take values in the range $[0,m)$.

Name	Popular Form	Canonical Form (P,N)	Mon.?	Range P,N	Rate $\eta(m)$
F_β-measure (Manning et al.)	$\frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP+\beta^2\cdot TP+FP}$	$\frac{(1+\beta^2)\cdot P}{\beta^2+\theta+P-\theta\cdot N}$	✓	$\left(0, 1+\frac{\beta^2}{\theta}\right)$	$\frac{m(1+\theta)}{m+\beta^2+\theta}$
Jaccard Coefficient (Koyejo et al.)	$\frac{TP}{TP+FP+FN}$	$\frac{P}{1+\theta-\theta\cdot N}$	✓	$\left(0, \frac{1+\theta}{\theta}\right)$	$\frac{m\theta}{1+\theta}$
Gower-Legendre$_{\sigma<1}$ (Sokolova & Lapalme)	$\frac{TP+TN}{TP+\sigma(FP+FN)+TN}$	$\frac{P+\theta\cdot N}{\sigma(1+\theta)+(1-\sigma)\cdot P+\theta(1-\sigma)\cdot N}$	✓	$(0,\infty)$	$\frac{(1-\sigma)m}{(1-\sigma)m+\sigma}$
Gower-Legendre$_{\sigma>1}$ (Sokolova & Lapalme)	$\frac{TP+TN}{TP+\sigma(FP+FN)+TN}$	$\frac{P+\theta\cdot N}{\sigma(1+\theta)+(1-\sigma)\cdot P+\theta(1-\sigma)\cdot N}$	✓	$\left(0, \frac{\sigma}{\sigma-1}\right)$	$\frac{(\sigma-1)m}{\sigma}$

Algorithm 2 AMP: Alternate Maximization Procedure

Input: Performance measure $\mathcal{P}_{(\mathbf{a},\mathbf{b})}$, feasible set $\mathcal{W}$, tolerance ϵ
Output: An ϵ-optimal classifier $\mathbf{w}\in\mathcal{W}$
1: Construct valuation function $V_{(\mathbf{a},\mathbf{b})}$
2: $\mathbf{w}_0 \leftarrow \mathbf{0}, t\leftarrow 1$
3: **while** $v_t > v_{t-1}+\epsilon$ **do**
4: $\quad\mathbf{w}_{t+1}\leftarrow \arg\max_{\mathbf{w}\in\mathcal{W}} V_{(\mathbf{a},\mathbf{b})}(\mathbf{w}, v_t)$
5: $\quad v_{t+1}\leftarrow \arg\max_{v>0} v$ such that $V_{(\mathbf{a},\mathbf{b})}(\mathbf{w}_{t+1}, v)\geq v$
6: $\quad t\leftarrow t+1$
7: **end while**
8: **return** $\mathbf{w}_t$

a classifier to achieve that level. The resulting algorithm **AMP** is detailed in Algorithm 2. Note that using Lemma 7, step 5 in the algorithm can be executed simply by setting $v_{t+1}=\mathcal{P}_{(\mathbf{a},\mathbf{b})}(\mathbf{w}_{t+1})$. Thus, in a very natural manner, the current classifier challenges the next classifier to beat its own performance. It turns out that this approach results in rapid convergence as outlined in the following theorem.

Theorem 8. *Let Algorithm 2 be executed with a performance measure $\mathcal{P}_{(\mathbf{a},\mathbf{b})}$ and reward functions that offer values in the range $[0,m)$. Let $\mathcal{P}^* := \sup_{\mathbf{w}\in\mathcal{W}}\mathcal{P}_{(\mathbf{a},\mathbf{b})}(\mathbf{w})$. Also let $\Delta_t = \mathcal{P}^* - \mathcal{P}_{(\mathbf{a},\mathbf{b})}(\mathbf{w}_t)$ be the excess error for the model $\mathbf{w}_t$ generated at time t. Then there exists a value $\eta(m)<1$ such that $\Delta_t \leq \Delta_0\cdot\eta(m)^t$.*

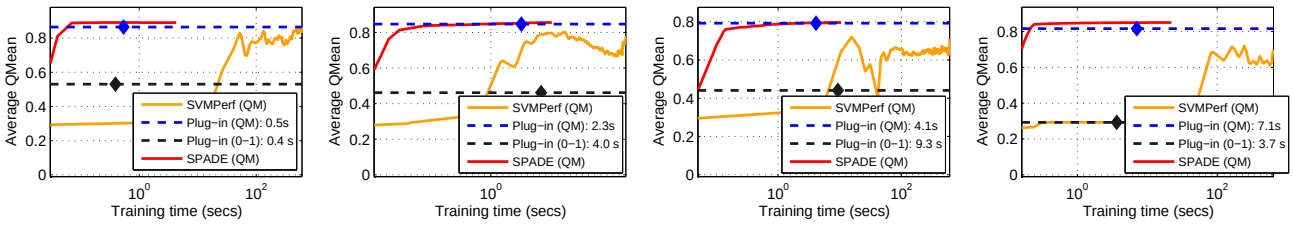
The proof of this theorem can be found in Appendix D. Table 2 gives values for the convergence rates of all the pseudo-linear performance measures, as well as the allowed range of values that the reward functions can take for those measures. This is important since performance measures such as the F-measure diverge if the reward function values approach 2. Other performance measures like the Gower-Legendre measure do not impose any such restrictions. Note that the above result shows that Algorithm 2 will always terminate in $\mathcal{O}\left(\log\frac{1}{\epsilon}\right)$ steps.

At this point it would be apt to make a historical note. Pseudo-linear functions have enjoyed a fair amount of interest in the optimization community (Schaible, 1976; Dinkelbach, 1967; Jagannathan, 1966) within the subfield of fractional programming. Of the many methods that have been developed to optimize these functions, the Dinkelbach-Jagannathan (DJ) procedure (Dinkelbach, 1967; Jagannathan, 1966) is of specific interest to us. It turns out that the **AMP** method can be seen as performing DJ-style updates over parameterized spaces (the parameter being the model $\mathbf{w}$). It is known (for instance see (Schaible, 1976)) that the DJ process is able to offer a linear convergence rates. Our proof of Theorem 8, which was obtained independently, can then be seen as giving a similar result in the parameterized setting.

Algorithm 3 STAMP: STochastic Alt-Max Procedure

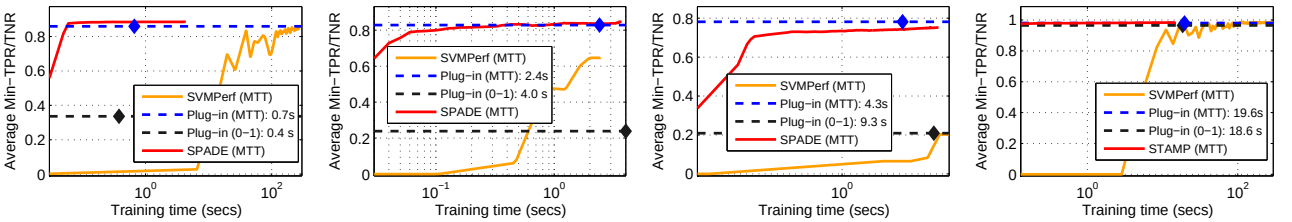
Input: Feasible set $\mathcal{W}$, Step sizes η_t, epoch lengths s_e, s'_e
Output: Classifier $\mathbf{w}\in\mathcal{W}$
1: $v\leftarrow 0, t\leftarrow 0, e\leftarrow 0, \mathbf{w}_0\leftarrow\mathbf{0}$
2: **repeat**
3: $\quad$*/* Model optimization stage */*
4: $\quad\widetilde{\mathbf{w}}\leftarrow\mathbf{w}_e$
5: $\quad$**while** $t<s_e$ **do**
6: $\quad\quad$Receive sample $(\mathbf{x},y)$
7: $\quad\quad\widetilde{\mathbf{w}}\leftarrow\widetilde{\mathbf{w}}+\eta_t\nabla_{\mathbf{w}}\left((1-\frac{v_e}{2})r^+(\widetilde{\mathbf{w}};\mathbf{x},y)+\frac{v_e}{2}r^-(\widetilde{\mathbf{w}};\mathbf{x},y)\right)$
8: $\quad\quad t\leftarrow t+1$
9: $\quad$**end while**
10: $\quad t\leftarrow 0, e\leftarrow e+1, \mathbf{w}_{e+1}\leftarrow\widetilde{\mathbf{w}}$
11: $\quad$*/* Challenge level estimation stage */*
12: $\quad v_+\leftarrow 0, v_-\leftarrow 0$
13: $\quad$**while** $t<s'_e$ **do**
14: $\quad\quad$Receive sample $(\mathbf{x},y)$
15: $\quad\quad v_y\leftarrow v_y + r^y(\mathbf{w}_e;\mathbf{x},y)$
16: $\quad\quad t\leftarrow t+1$
17: $\quad$**end while**
18: $\quad t\leftarrow 0, v_e\leftarrow\frac{2v_+}{2+v_+-v_-}$
19: **until** stream is exhausted
20: **return** $\mathbf{w}_e$

However, we wish to move one step further and optimize these performance measures in an online stochastic manner. To this end, we observe that the **AMP** algorithm can be executed in an online fashion by using stochastic updates to train the intermediate models. The resulting algorithm **STAMP**, is presented in Algorithm 3. However, this algorithm is much harder to analyze because unlike **AMP** which has the luxury of offering exact updates, **STAMP** offers inexact, even noisy updates. Indeed, even existing works in the optimization community (for example (Schaible, 1976)) do not seem to have analyzed DJ-style methods with noisy updates.



(a) IJCNN1 (b) KDD08 (c) PPI (d) Covtype

Figure 1. Comparison of stochastic primal-dual method (SPADE) with baseline methods on QMean maximization tasks. SPADE achieves similar/better accuracies while consistently requiring about 3-4x less time than other baseline approaches.



(a) IJCNN1 (b) KDD08 (c) PPI (d) MNIST

Figure 2. Comparison of stochastic primal-dual method (SPADE) with baseline methods on Min-TPR/TNR maximization tasks

Our next contribution hence, is an analysis of the convergence rate offered by the **AMP** algorithm when neither of the two maximizations is carried out exactly. For the sake of simplicity, we present the **STAMP** algorithm and its analysis for the case of F_1 measure. Suppose at each time step, for some $\epsilon_t \geq 0, \delta_t$, we have

$$V(\mathbf{w}_{t+1}, v_t) = \max_{\mathbf{w} \in \mathcal{W}} V(\mathbf{w}, v_t) - \epsilon_t$$
$$v_t = F(\mathbf{w}_t) + \delta_t,$$

then for some $\eta < 1$, we have

$$\Delta_T \leq \eta^T \Delta_0 + \sum_{i=0}^{T-1} \eta^{T-i} \left(|\delta_t| + \epsilon_t \right)$$

As a corollary we present a convergence analysis for the **STAMP** algorithm in Theorem 9.

Theorem 9. *Let Algorithm 3 be executed with a performance measure $\mathcal{P}_{(\mathbf{a},\mathbf{b})}$ and reward functions with range $[0, m)$. Let $\eta = \eta(m)$ be the rate of convergence guaranteed for $\mathcal{P}_{(\mathbf{a},\mathbf{b})}$ by the **AMP** algorithm. Set the epoch lengths to $s_e, s'_e = \widetilde{\mathcal{O}}\left(\frac{1}{\eta^{2e}}\right)$. Then after $e = \log_{\frac{1}{\eta}} \left(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon} \right)$ epochs, we can ensure with probability at least $1-\delta$ that $\mathcal{P}^* - \mathcal{P}_{(\mathbf{a},\mathbf{b})}(\mathbf{w}_e) \leq \epsilon$. Moreover the number of samples consumed till this point is at most $\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon^2}\right)$.*

The convergence analysis for noisy **AMP** can be found in Appendix E. The proof of this theorem can be found in Appendix F. Both results require a fine grained analysis of how errors accumulate throughout the learning process.

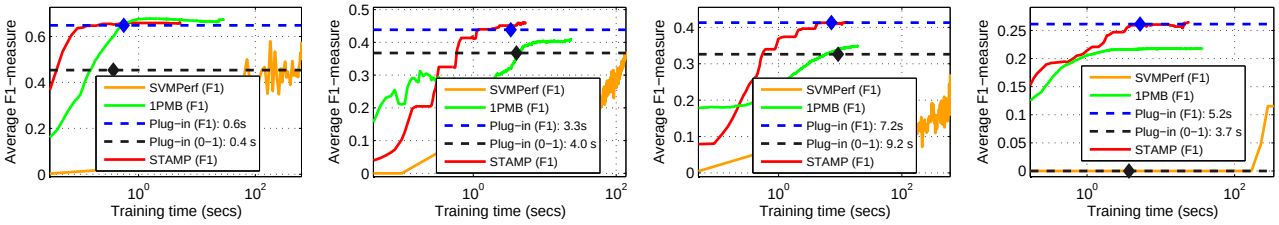
6. Experimental Results

We shall now compare our methods with the state-of-the-art on various performance measures and datasets.

Datasets: We evaluated our methods on 5 publicly available benchmark datasets: a) PPI, b) KDD Cup 2008, c) IJCNN, d) Covertype, e) MNIST. All datasets exhibited moderate to severe label imbalance with the KDD Cup 2008 dataset having just 0.61% positives.

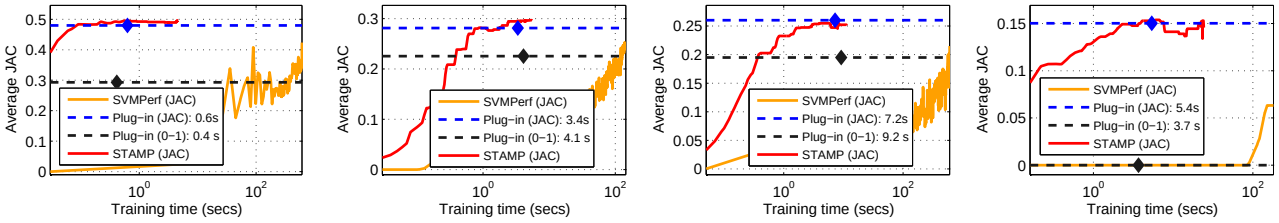
Methods: We instantiated the **SPADE** algorithm (Algorithm 1) on the Q-mean and Min-TPR/TNR performance measures. We also instantiated the **STAMP** method (Algorithm 3) on F1-measure and the JAC coefficient. In both cases we compared to the SVMPerf method (Joachims et al., 2009) and plug-in method (Koyejo et al., 2014) specialized to these measures. For the sake of reference, we also compared to the standard logistic regression method for (unweighted) binary classification. Additionally for F1-measure, we also compared to the **1PMB** stochastic gradient descent method proposed recently by (Kar et al., 2014). All methods were implemented in C.

Parameters: We used 70% of the dataset for training and the rest for testing. Tunable parameters, including thresholds for the plug-in approaches, were cross-validated on a



(a) IJCNN1 (b) KDD08 (c) PPI (d) Covtype

Figure 3. Comparison of stochastic alternating minimization procedure (STAMP) with baseline methods on F1 maximization tasks



(a) IJCNN1 (b) KDD08 (c) PPI (d) Covtype

Figure 4. Comparison of stochastic alternating minimization procedure (STAMP) with baseline methods on JAC maximization tasks

validation set. All results reported here were averaged over 5 random train-test splits. We used hinge-loss based reward functions for our methods. **STAMP** was executed by setting the challenge level to the actual F-measure/JAC at each stage. We used a state of the art LBFGS solver to implement the plug-in methods and used standard implementations of the SVMPerf algorithm. Since our methods are able to take a single pass over the data very rapidly, **SPADE** was allowed to run for 25 passes over the data and **STAMP** was allowed 25 passes with an initial epoch length of 100 which was doubled after every iteration. The SVMPerf algorithm was allowed a runtime of up to $50\times$ of that given to our method after which it was terminated. The LBFGS solver was always allowed to run till convergence.

Figures 1 and 2 compare the **SPADE** method with the baseline methods for the Q-mean and Min-TPR/TNR measures. In general, **SPADE** was found to offer comparable or superior accuracies with greatly accelerated convergence as compared to other methods. On the IJCNN and Covtype datasets, **SPADE** outperformed every other method by about 2-3%. As **SPADE** is a stochastic first order method, it is expected to rapidly find out a fairly accurate solution. Indeed, the method was found to offer greatly accelerated convergence without fail. For instance, on the MNIST dataset, **SPADE** found out the best solution as much as $60\times$ faster than any other method whereas on the KDD Cup and PPI datasets it was $12\times$ and $2\times$ faster respectively. The SVMPerf method, on the other hand, was found to be extremely slow in general and require at least an order of magnitude time more than **SPADE** to find reasonably accurate solutions. It is also notable that in all cases, simple binary classification gave very poor accuracies due to the severe label imbalance in these datasets.

Figures 3 and 4 report the performance of the **STAMP** method applied to pseudo-linear functions. Similar to the concave measures, **STAMP** was found to provide competitive accuracies as compared to the baseline methods but require at least $3-4\times$ less computational time. Interestingly, for the F1-measure, the **1PMB** method, which is another stochastic gradient descent-based method, was found to struggle to obtain accuracies similar to that of **STAMP** or else offer much slower convergence. We suspect two main reasons for the suboptimal behavior of this other stochastic method. Firstly these results confirm the adverse effect of the dependence on an in-memory buffer on these methods. It is notable that this dependence causes even the theoretical convergence rates for these methods to be weaker as was noted earlier in the discussion. Secondly, we note that both SVMPerf and **1PMB** optimize the same "struct-SVM" style surrogate for the F-measure (Kar et al., 2014). This surrogate has been observed to give poor accuracies when compared to plug-in methods in several previous works (Koyejo et al., 2014; Narasimhan et al., 2014). **STAMP** on the other hand, works directly with F-measure in a manner similar to, but faster than, the plug-in methods which might explain its better performance.

Acknowledgements

HN thanks support from a Google India PhD Fellowship.

References

Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

Cesa-Bianchi, Nicoló, Conconi, Alex, and Gentile, Claudio. On the Generalization Ability of On-Line Learning Algorithms. In *15th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 359–366, 2001.

Daskalaki, Sophia, Kopanas, Ioannis, and Avouris, Nikolaos. Evaluation of Classifiers for an Uneven Class Distribution Problem. *Applied Artificial Intelligence*, 20: 381–417, 2006.

Dembczyński, Krzysztof, Jachnik, Arkadiusz, Kotlowski, Wojciech, Waegeman, Willem, and Hüllermeier, Eyke. Optimizing the F-Measure in Multi-Label Classification: Plug-in Rule Approach versus Structured Loss Minimization. In *30th International Conference on Machine Learning (ICML)*, 2013.

Dinkelbach, Werner. On Nonlinear Fractional Programming. *Management Science*, 13(7, Series A, Sciences): 492–498, 1967.

Hsu, Daniel, Kakade, Sham, Langford, John, and Zhang, Tong. Multi-Label Prediction via Compressed Sensing. In *23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 772–780, 2009.

Jagannathan, R. On Some Properties of Programming Problems in Parametric Form Pertaining to Fractional Programming. *Management Science*, 12(7, Series A, Sciences):609–615, 1966.

Jaggi, Martin, Smith, Virginia, Takác, Martin, Terhorst, Jonathan, Krishnan, Sanjay, Hofmann, Thomas, and Jordan, Michael I. Communication-Efficient Distributed Dual Coordinate Ascent. In *28th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 694–702, 2014.

Joachims, Thorsten, Finley, Thomas, and Yu, Chun-Nam John. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

Kar, Purushottam, Sriperumbudur, Bharath K, Jain, Prateek, and Karnick, Harish. On the Generalization Ability of Online Learning Algorithms for Pairwise Loss Functions. In *30th International Conference on Machine Learning (ICML)*, 2013.

Kar, Purushottam, Narasimhan, Harikrishna, and Jain, Prateek. Online and Stochastic Gradient Methods for Non-decomposable Loss Functions. In *28th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 694–702, 2014.

Kennedy, Kenneth, Namee, Brian Mac, and Delany, Sarah Jane. Learning without default: a study of one-class classification and the low-default portfolio problem. In *International Conference on Artificial Intelligence and Cognitive Science (ICAICS)*, volume 6202 of *Lecture Notes in Computer Science*, pp. 174–187, 2010.

Koyejo, Oluwasanmi O., Natarajan, Nagarajan, Ravikumar, Pradeep K., and Dhillon, Inderjit S. Consistent Binary Classification with Generalized Performance Metrics. In *28th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 2744–2752, 2014.

Lewis, D.D. Evaluating and optimizing autonomous text classification systems. In *18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1995.

Liu, W. and Chawla, S. A Quadratic Mean based Supervised Learning Model for Managing Data Skewness. In *11th SIAM International Conference on Data Mining (SDM)*, 2011.

Mahdavi, Mehrdad, Yang, Tianbao, and Jin, Rong. Stochastic Convex Optimization with Multiple Objectives. In *27th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1115–1123, 2013.

Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Narasimhan, Harikrishna, Vaish, Rohit, and Agarwal, Shivani. On the Statistical Consistency of Plug-in Classifiers for Non-decomposable Performance Measures. In *28th Annual Conference on Neural Information Processing Systems (NIPS)*, 2014.

Parambath, Shameem Puthiya, Usunier, Nicolas, and Grandvalet, Yves. Optimizing F-Measures by Cost-Sensitive Classification. In *28th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 2123–2131, 2014.

Rakhlin, Alexander, Sridharan, Karthik, and Tewari, Ambuj. Online Learning: Beyond Regret. In *24th Annual Conference on Learning Theory (COLT)*, 2011.

Schaible, Siegfried. Fractional Programming. II, on Dinkelbach's Algorithm. *Management Science*, 22(8): 868–873, 1976.

Shalev-Shwartz, Shai, Singer, Yoram, Srebro, Nathan, and Cotter, Andrew. Pegasos: primal estimated sub-gradient solver for SVM. *Math. Program.*, 127(1):3–30, 2011.

Sokolova, Marina and Lapalme, Guy. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

Vincent, P.H. *An Introduction to Signal Detection and Estimation*. Springer-Verlag New York, Inc., 1994.

Ye, Nan, Chai, Kian Ming A., Lee, Wee Sun, and Chieu, Hai Leong. Optimizing F-Measures: A Tale of Two Approaches. In *29th International Conference on Machine Learning (ICML)*, 2012.

Zinkevich, Martin. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *20th International Conference on Machine Learning (ICML)*, pp. 928–936, 2003.