AP-Perf: Incorporating Generic Performance Metrics in Differentiable Learning

Rizal Fathony
Carnegie Mellon University
rfathony@cs.cmu.edu

J. Zico Kolter
Carnegie Mellon University and Bosch Center for AI
zkolter@cs.cmu.edu

Abstract

We propose a method that enables practitioners to conveniently incorporate custom non-decomposable performance metrics into differentiable learning pipelines, notably those based upon neural network architectures. Our approach is based on the recently developed adversarial prediction framework, a distributionally robust approach that optimizes a metric in the worst case given the statistical summary of the empirical distribution. We formulate a marginal distribution technique to reduce the complexity of optimizing the adversarial prediction formulation over a vast range of non-decomposable metrics. We demonstrate how easy it is to write and incorporate complex custom metrics using our provided tool. Finally, we show the effectiveness of our approach various classification tasks on tabular datasets from the UCI repository and benchmark datasets, as well as image classification tasks. The code for our proposed method is available at https://github.com/rizalzaf/AdversarialPrediction.jl.

1 INTRODUCTION

In real-world applications, the performance of machine learning algorithms is measured with evaluation metrics specifically tailored to the problem of interest. Although the accuracy is the most popular evaluation metric, many applications require the use of more complex evaluation metrics that are not additively decomposable into sample-wise measures. For example, in text classification area, F_β score (weighted harmonic mean of precision and recall) is frequently used to evaluate the performance. F_β is also popular in the classification tasks with imbalanced datasets. In medical fields, the sensitivity, specificity, and informedness are some of the popular evaluation metrics. Many of these performance metrics require inherent trade-offs, for example, balancing precision versus recall.

A variety of learning algorithms that incorporate some of the performance metrics above into their learning objectives have been proposed. One of the first approaches to this problem is the SVM-Perf (Joachims, 2005), which augments the constraints of a binary SVM optimization with the metrics. Koyejo et al. (2014) and Narasimhan et al. (2014) proposed plug-in classifiers that rely on an external estimator of class probability (typically using logistic regression). Hazan et al. (2010) proposed a way to directly optimizes the performance metric by computing the asymptotic gradient of the metric. Some of the previous research focused only on a specific performance metric, most notably, the F1-score (Dembczynski et al., 2011; Parambath et al., 2014; Lipton et al., 2014; Wang et al., 2015; Shi et al., 2017). Optimizing the metric on specific learning settings have also been explored, for example, in online learning (Busa-Fekete et al., 2015; Kar et al., 2014; Narasimhan et al., 2015) and ranking (Yue et al., 2007; Narasimhan and Agarwal, 2013a,b; Kar et al., 2015). Finally, several efforts have been made to incorporate non-decomposable metrics into neural networks training (Eban et al., 2017; Song et al., 2016; Sanyal et al., 2018).

Despite this rich literature on learning algorithms for non-decomposable metrics, the algorithms have not been widely used in practical applications, particularly in the modern machine learning applications that rely on the representational power of neural network architectures, where training is typically done using a gradient-based method. Instead of being trained to optimize the evaluation metric of interest, they are typically trained to minimize cross-entropy loss, with the hope that it will indirectly optimize the metric (Eban et al., 2017). However, as mentioned in previous re-

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

```
model = Chain(
    Dense(nvar, 100, relu),
    Dense(100, 100, relu),
    Dense(100, 1),
    vec)

objective(x, y) = mean(
  logitbinarycrossentropy(model(x), y))

opt = ADAM(1e-3)
Flux.train!(objective, params(model),
  train_set, opt)
```

```
model = Chain(Dense(nvar, 100, relu),
    Dense(100, 100, relu), Dense(100, 1), vec)

@metric FBeta beta
function define(::Type{FBeta}, C::ConfusionMatrix, beta)
    return ((1 + beta^2) * C.tp) / (beta^2 * C.ap + C.pp)
end
f2_score = FBeta(2)
special_case_positive!(f2_score)

objective(x, y) = ap_objective(model(x), y, f2_score)
Flux.train!(objective, params(model), train_set, ADAM(1e-3))
```

Figure 1: Code examples for incorporating F_2-score metric into a neural network training pipeline (right), compared with the standard code for cross-entropy objective (left). The codes are implemented in Julia.

search (Cortes and Mohri, 2004; Davis and Goadrich, 2006), this discrepancy between the target and optimized metric may lead to inferior results.

We argue that two factors hinder the wide adoption of the learning algorithms for non-decomposable metrics into many modern machine learning applications. First, many of the existing learning algorithms are not flexible enough to accommodate the custom need of real-world applications. Their formulations only cover a few types of performance metrics that may not be relevant for some applications. Second, even though some of the existing formulations are flexible, they do not provide a way for practitioners to customize the usage. The authors of these flexible methods oftentimes only provide few uses case metrics in their experiments and also their published codes. A significant amount of effort (e.g., deriving the formulations and rewriting the codes) need to be spent by a practitioner who wants to implement and customize their algorithm to the specific needs of the applications. This also still be a problem even for the latest development of algorithms that already specifically target neural network training. These two factors force many practitioners to choose a method that is easy to incorporate to their machine learning system, for example, the cross-entropy objective (a common proxy for accuracy metric) that is readily available in many frameworks.

In this paper, our goal is to overcome the problem above. We propose a generic framework for optimizing arbitrary complex non-decomposable performance metrics using gradient-based learning procedures. Our framework can be applied to most of the common use-cases of non-decomposable metrics. Specifically, we require the metric to be derived from the value of the confusion matrix with minimal requirements on how the metric needs to be constructed. Our formulation also supports optimizing a performance metric with a constraint over another metric. This is useful in the case where we want to balance the trade-off between two metrics, for example, in the case where we want to maximize precision subject to recall ≥ 0.8. Our approach is based on the adversarial prediction framework (Fathony et al., 2018a; Asif et al., 2015), a distributionally robust framework for constructing learning algorithms that seeks a predictor that maximizes the performance metric in the worst case given the statistical summary of the empirical distribution. We replace the empirical data for evaluating the predictor with an adversary that is free to choose an evaluating distribution from the set of conditional distribution that matches the statistics of empirical data via moment matching on the features. Although naively applying this approach is not possible, we develop a marginalization technique that reduces the number of variables in the resulting optimization from exponentially many variables to just quadratic.

In addition to these algorithmic contributions, we establish the Fisher consistency of the method, a feature notably lacking from much past work approximately optimizing performance metrics (Tewari and Bartlett, 2007; Liu, 2007). We also develop a programming interface such that a practitioner can easily construct the metric and integrate it into their learning pipeline. Figure 1 provides an example of incorporating the F_2-score metric into the training pipeline of our method. Notice that only minimal changes from the standard cross-entropy learning code are needed. Finally, we evaluate the performance of our method against the standard training on several benchmark datasets within neural network learning pipelines and demonstrate that our method vastly outperforms traditional approaches for training these networks.

2 BACKGROUND

2.1 Performance Metrics

Deciding on what performance metric to be used for evaluating the prediction is an important aspect of machine learning applications, since it will also guide the

Table 1: Confusion Matrix

		Actual		
		Positive	Negative	
Pred.	Positive	True Pos. (TP)	False Pos. (FP)	Predicted Pos. (PP)
	Negative	False Neg. (FN)	True Neg. (TN)	Predicted Neg. (PN)
		Actual Pos. (AP)	Actual Neg. (AN)	All Data (ALL)

design of learning algorithms. A performance metric should be carefully picked to reflect the objective goal of the prediction (Powers, 2011). Different tasks in machine learning require different metrics that align well with the tasks. For binary classification problems, many of the commonly used performance metrics are derived from the confusion matrix. The confusion matrix is a table that reports the values that relate the prediction of a classifier with the ground truth labels. Table 1 shows the anatomy of the confusion matrix.

Most commonly used performance metrics can be derived from the confusion matrix. Some of the metrics are decomposable, which means that it can be broken down to an independent sum of another metric that depends only on a single sample. However, most of the interesting performance metrics are non-decomposable, where we need to consider all samples at once. There is a wide variety of non-decomposable performance metrics. Table 2 shows some of the popular metrics and the formula on how to derive the metric from the confusion matrix.

2.2 Existing Methods

Many existing methods have been proposed for optimizing non-decomposable metrics. However, they do not facilitate an easy way to implement the methods on new custom tasks. They also do not provide convenient ways to integrate the algorithms to differentiable learning pipeline on custom non-decomposable performance metrics. SVM-Perf (Joachims, 2005) is a large margin technique that enables the incorporation of a performance metric to the SVM training objective. However, for new metrics that are not explained in the paper, we need to formulate and implement an algorithm to find the maximum violated constraints for the given metric inside its cutting plane algorithm. Plug-in based classifiers (Koyejo et al., 2014; Narasimhan et al., 2014) need to first solve probability estimation problems optimally, and then tune a threshold depending on the performance metric they optimize. This makes the techniques hard to incorporate into differentiable learning pipelines. Many existing methods only focus on developing formulations for specific performance metrics or providing examples on a few metrics without any complete guide on extending the methods to other metrics (Hazan et al., 2010; Dembczynski et al., 2011; Parambath et al., 2014; Lipton et al., 2014; Busa-Fekete et al., 2015; Wang et al., 2015; Shi et al., 2017). Finally, even though some of the existing methods (Eban et al., 2017; Song et al., 2016; Sanyal et al., 2018) specifically targeted their approach to neural network learning, they do not provide an easy way to implement their method on new custom metrics.

Table 2: Examples of Non-Decomposable Performance Metrics

NAME	FORMULA
F_β-score	$\frac{(1+\beta^2)\,\mathsf{TP}}{\beta^2\,\mathsf{AP}+\mathsf{PP}}$
Geom. mean of Prec. & Recall	$\frac{\mathsf{TP}}{\sqrt{\mathsf{PP}\cdot\mathsf{AP}}}$
Balanced Accuracy	$\frac{1}{2}\left(\frac{\mathsf{TP}}{\mathsf{AP}}+\frac{\mathsf{TN}}{\mathsf{AN}}\right)$
Bookmaker Informedness	$\frac{\mathsf{TP}}{\mathsf{AP}}+\frac{\mathsf{TN}}{\mathsf{AN}}-1$
Cohen's kappa score	$\frac{(\,\mathsf{TP}+\mathsf{TN}\,)/\,\mathsf{ALL}-(\,\mathsf{AP}\cdot\mathsf{PP}+\mathsf{AN}\cdot\mathsf{PN}\,)/\,\mathsf{ALL}^{\,2}}{1-(\,\mathsf{AP}\cdot\mathsf{PP}+\mathsf{AN}\cdot\mathsf{PN}\,)/\,\mathsf{ALL}^{\,2}}$
Matthews correlation coefficient	$\frac{\mathsf{TP}\,/\,\mathsf{ALL}-(\,\mathsf{AP}\cdot\mathsf{PP}\,)/\,\mathsf{ALL}^{\,2}}{\sqrt{\mathsf{AP}\cdot\mathsf{PP}\cdot\mathsf{AN}\cdot\mathsf{PN}\,/\,\mathsf{ALL}^{\,2}}}$

2.3 Adversarial Prediction

Recently developed adversarial prediction framework (Fathony et al., 2018a; Asif et al., 2015) provides an alternative to the empirical risk minimization framework (ERM) (Vapnik, 1992) for designing learning algorithms. In a classification setting, the ERM framework prescribes the use of convex surrogate loss function as a tractable approximation to the original non-convex and non-continuous objective of optimizing an evaluation metric (e.g., accuracy). In contrast, the adversarial prediction framework replaces the empirical training data for evaluating the metric with an adversary that is free to choose an evaluating distribution that approximates the training data. This approximation of the training data is performed by constraining the adversary's distribution to match the feature statistics of the empirical training data. Even though we started with a non-convex and non-continuous metric, the resulting optimization objective is always convex with respect to the optimized variable.

The adversarial prediction framework has been previously used to design learning algorithms for many decomposable metrics, including the zero-one loss (Fathony et al., 2016), ordinal regression loss (Fathony et al., 2017), abstention loss (Fathony et al., 2018a), cost-sensitive loss metrics (Asif et al., 2015). The extensions of the framework to non-decomposable metrics and structured prediction have also been explored.

The main challenge of these extensions is that naively solve the resulting dual optimization is intractable since we have to simultaneously consider all possible label assignments for all samples in the dataset. Previous research have tried to reduce the complexity of solving the problem. One of the first efforts by Wang et al. (2015) uses a double oracle technique to solve the problem for a few performance metrics (F_1-score, precision@k, and DCG). However, the double oracle algorithm they use does not have any guarantee that it will converge to the solution in polynomial time. Additionally, extending the approach to other metrics is hard since we have to formulate an algorithm to find the best response for the given metric, which is harder than the SVM-Perf's problem on finding the most violated constraint.

The second wave of research have been proposed for applying the adversarial prediction to non-decomposable metrics and structured prediction using marginalization technique that reduces the optimization over full exponentially sized conditional distributions into their polynomially sized marginal distributions. This technique has been applied to the problem of optimizing the F_1-score metric (Shi et al., 2017), tree-structured graphical models (Fathony et al., 2018c), and bipartite matching in graphs (Fathony et al., 2018b). However, these methods only focus on the specific performance metrics, and they do not provide a way to extend the method to custom performance metrics easily. Our paper is the first effort to generalize the marginalization technique to a vast range of performance metrics. Our approach is also the first method that can be easily integrated into differentiable learning pipelines.

3 APPROACH

To achieve our goal of providing a flexible and easy to use method for optimizing custom performance metrics, we formulate it as an adversarial prediction task.

3.1 Adversarial Prediction Formulation

In a binary classification task, the training examples consist of pairs of training data and labels $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ drawn i.i.d from a distribution D on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the feature space and $\mathcal{Y} = \{0, 1\}^n$ is the set of binary labels. A classifier needs to make a prediction $\hat{y}_i$ for each sample $\mathbf{x}_i$. The prediction is evaluated using a non-decomposable performance metric, $\text{metric}(\hat{\mathbf{y}}, \mathbf{y})$. Here, we need to consider the prediction for all samples (denoted in vector notations) to compute the metric.

The adversarial prediction method seeks a predictor that robustly maximizes the performance metric against an adversary that is constrained to approximate the training data (via moment matching constraints on the features) but otherwise aims to minimize the metric. Both predictor and adversary players are allowed to make probabilistic predictions over all possible label outcomes. Denote $\mathcal{P}(\hat{\mathbf{Y}}) \triangleq \hat{P}(\hat{\mathbf{Y}}|\mathbf{X})$ as the predictor's probabilistic prediction and $\mathcal{Q}(\check{\mathbf{Y}}) \triangleq \check{P}(\check{\mathbf{Y}}|\mathbf{X})$ as the adversary's distribution.[1] The adversary player needs to approximate the training data by selecting a conditional probability $\mathcal{Q}(\check{\mathbf{Y}})$ whose feature expectations match the empirical feature statistics. On the other hand, the predictor is free to choose any conditional probability $\mathcal{P}(\hat{\mathbf{Y}})$ that maximizes the expected metric. Formally, the adversarial prediction is formulated as:

$$\max_{\mathcal{P}(\hat{\mathbf{Y}})} \min_{\mathcal{Q}(\check{\mathbf{Y}})} \mathbb{E}_{\tilde{P}(\mathbf{X});\mathcal{P}(\hat{\mathbf{Y}});\mathcal{Q}(\check{\mathbf{Y}})} \left[\text{metric}(\hat{\mathbf{Y}}, \check{\mathbf{Y}})\right]$$
$$\text{s.t.: } \mathbb{E}_{\tilde{P}(\mathbf{X});\mathcal{Q}(\check{\mathbf{Y}})}[\phi(\mathbf{X}, \check{\mathbf{Y}})] = \mathbb{E}_{\tilde{P}(\mathbf{X},\mathbf{Y})}\left[\phi(\mathbf{X}, \mathbf{Y})\right], \quad (1)$$

where $\tilde{P}$ denotes the empirical distribution. Using the method of Lagrangian multipliers and strong duality for convex-concave saddle point problems (Von Neumann and Morgenstern, 1945; Sion, 1958), the dual formulation of Eq. (1) can be written as:

$$\max_{\theta} \mathbb{E}_{\tilde{P}(\mathbf{X},\mathbf{Y})} \Bigg[\min_{\mathcal{Q}(\check{\mathbf{Y}})} \max_{\mathcal{P}(\hat{\mathbf{Y}})} \mathbb{E}_{\mathcal{P}(\hat{\mathbf{Y}});\mathcal{Q}(\check{\mathbf{Y}})} \Big[\text{metric}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) - \theta^\intercal \left(\phi(\mathbf{X}, \check{\mathbf{Y}}) - \phi(\mathbf{X}, \mathbf{Y})\right) \Big] \Bigg], \quad (2)$$

where θ is the Lagrange dual variable for the moment matching constraints of the adversary's distribution. This follows directly from previous results in adversarial prediction (Fathony et al., 2018a).

3.2 Adversarial Prediction for Non-Decomposable Performance Metrics

We consider a family of performance metrics that can be expressed as a sum of fractions of the entities in the confusion matrix (Table 1):

$$\text{metric}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_j \frac{a_j \mathsf{TP} + b_j \mathsf{TN} + f_j(\mathsf{PP}, \mathsf{AP})}{g_j(\mathsf{PP}, \mathsf{AP})}, \quad (3)$$

where a_j and b_j are constants, whereas f_j and g_j are functions over PP and AP. Hence, the numerator is a linear function over true positive (TP) and true negative (TN) which may also depends on sum statistics, i.e., predicted and actual positive (PP and AP) as well as their negative counterparts (predicted and actual negative (PN and AN)) and all data (ALL)[2]. The denominator depends only on the sum statistics. This

[1] Lowercase y and $\mathbf{y}$, denote scalar and vector values, and capitals, Y or $\mathbf{Y}$, denote random variables.

[2] We simplify the inputs of f_j and g_j to be just PP and AP since the other terms can be derived from PP and AP. ALL is just a constant, whereas $\mathsf{PN} = \mathsf{ALL} - \mathsf{PP}$ and $\mathsf{AN} = \mathsf{ALL} - \mathsf{AP}$.

metric construction covers a vast range of commonly used metrics, including all metrics in Table 2.

Applying the adversarial prediction framework to classification problems with non-decomposable metrics is non-trivial. We take a look at the inner minimax problem of the dual formulation (Eq. (2)), i.e.:

$$\min_{\mathcal{Q}(\check{\mathbf{Y}})} \max_{\mathcal{P}(\hat{\mathbf{Y}})} \mathbb{E}_{\mathcal{P}(\hat{\mathbf{Y}});\mathcal{Q}(\check{\mathbf{Y}})} \left[\text{metric}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) - \theta^\intercal \phi(\mathbf{X}, \check{\mathbf{Y}})\right]. \tag{4}$$

Note that we set aside the empirical potential term ($\theta^\intercal \phi(\mathbf{X}, \mathbf{Y})$) since it does not influence the inner minimax solution. Unlike many previous adversarial prediction research (Asif et al., 2015; Fathony et al., 2016, 2017, 2018a), we cannot reduce the problem to sample-wise minimax problems since our metric is now non-decomposable. We need to deal with the full conditional distribution ($\mathcal{P}(\hat{\mathbf{y}})$ and $\mathcal{Q}(\check{\mathbf{y}})$) over all samples which is exponential in size. Therefore, naively solving the inner minimax problem is intractable. In the subsequent analyses, we aim to reduce the complexity of solving the problem by optimizing over the marginal distribution of $\mathcal{P}(\hat{\mathbf{y}})$ and $\mathcal{Q}(\check{\mathbf{y}})$.

We take a look at the expectation of the metric. We now define the marginal probability of the event where $y_i = 1$ and $\sum_{i'} y_{i'} = k$, which we write as $\mathcal{P}(\hat{y}_i = 1, \sum_{i'} \hat{y}_{i'} = k))$ and $\mathcal{Q}(\check{y}_i = 1, \sum_{i'} \check{y}_{i'} = k))$ for the predictor and adversary respectively. Similarly, we also define the marginal probability of the event where $y_i = 0$ and $\sum_{i'} y_{i'} = k$. Let us denote $\mathbf{p}_k^a$ be a vector with n items where each of its items $(\mathbf{p}_k^a)_i$ represents the predictor's marginal probability $\mathcal{P}(\hat{y}_i = a, \sum_{i'} \hat{y}_{i'} = k))$. Similarly, we also denote $\mathbf{q}_l^a$ for the adversary's marginal probabilities. We also denote the marginal probability of sums as $r_k = \mathcal{P}(\sum_i \hat{y}_i = k)$, and $s_l = \mathcal{Q}(\sum_i \check{y}_i = l)$ Using these notations, we simplify the computation of the expected value of the performance metric in terms of these marginal probabilities as stated in Theorem 1.[3]

Theorem 1. *Given a performance metric that follows the construction in Eq.* (3)*, the expected value of the metric over exponentially sizes conditional probabilities $\mathcal{P}(\hat{\mathbf{Y}})$ and $\mathcal{Q}(\check{\mathbf{Y}})$ can be expressed as the sum of functions over marginal probability variables $\mathbf{p}_k^1$, $\mathbf{q}_l^1$, $\mathbf{p}_k^0$, $\mathbf{q}_l^0$, r_k, and s_l as follows:*

$$\mathbb{E}_{\mathcal{P}(\hat{\mathbf{Y}});\mathcal{Q}(\check{\mathbf{Y}})} \left[\textit{metric}(\hat{\mathbf{Y}}, \check{\mathbf{Y}})\right] = \sum_{k\in[0,n]} \sum_{l\in[0,n]} \sum_j \frac{1}{g_j(k,l)} \big\{ a_j[\mathbf{p}_k^1 \cdot \mathbf{q}_l^1] + b_j[\mathbf{p}_k^0 \cdot \mathbf{q}_l^0] + f_j(k,l) r_k s_l \big\}. \tag{5}$$

Some performance metrics (e.g. precision, recall, F-score, sensitivity, and specificity) enforce special cases to avoid division by zero. For the metrics that contains true positive, the special cases is usually defined as:

$$\begin{aligned} &\text{metric}(\mathbf{0}, \mathbf{0}) = 1; \quad \text{metric}(\mathbf{0}, \mathbf{y}) = 0, \forall \mathbf{y} \neq \mathbf{0}; \\ &\text{metric}(\hat{\mathbf{y}}, \mathbf{0}) = 0, \forall \hat{\mathbf{y}} \neq \mathbf{0}, \end{aligned} \tag{6}$$

whereas for the one with true negative, their cases are:

$$\begin{aligned} &\text{metric}(\mathbf{1}, \mathbf{1}) = 1; \quad \text{metric}(\mathbf{1}, \mathbf{y}) = 0, \forall \mathbf{y} \neq \mathbf{1}; \\ &\text{metric}(\hat{\mathbf{y}}, \mathbf{1}) = 0, \forall \hat{\mathbf{y}} \neq \mathbf{1}. \end{aligned} \tag{7}$$

Here $\hat{\mathbf{y}} = \mathbf{0}$ and $\hat{\mathbf{y}} = \mathbf{1}$ means that the classifier predicts all samples as negative and positive respectively. If the special cases are enforced, we need to modify Eq. (5) accordingly. For example, if both special cases for true positive and true negative are enforced, it becomes:

$$\sum_{k\in[1,n-1]} \sum_{l\in[1,n-1]} \sum_j \frac{1}{g_j(k,l)} \big\{ a_j[\mathbf{p}_k^1 \cdot \mathbf{q}_l^1] + b_j[\mathbf{p}_k^0 \cdot \mathbf{q}_l^0] + f_j(k,l) r_k s_l \big\} + \mathcal{P}(\mathbf{0})\mathcal{Q}(\mathbf{0}) + \mathcal{P}(\mathbf{1})\mathcal{Q}(\mathbf{1}). \tag{8}$$

Let us denote a $n \times n$ marginal distribution matrix $\mathbf{P}$ where each column $\mathbf{P}_{(:,k)}$ represents $\mathbf{p}_k^1$. Similarly, we denote a matrix $\mathbf{Q}$ for $\mathbf{q}_k^1$. For our feature, we use additive feature function, i.e., $\phi(\mathbf{x}, \mathbf{y}) = \sum_i \phi(\mathbf{x}_i, y_i)$. For simplicity, we also define $\phi(\mathbf{x}_i, y_i = 0) = 0$. Let us denote Ψ be a $n \times m$ matrix where each of its columns denotes the feature for each sample, i.e., $\Psi_{:,i} = \phi(\mathbf{x}_i, y_i = 1)$, and m is the number of features. Using these notations, we simplify the dual formulation of the adversarial prediction in Theorem 2.

Theorem 2. *Let $\mathbf{P}$ and $\mathbf{Q}$ be the marginal predictor and adversary probability matrices respectively. Given a performance metric that follows the construction in Eq.* (3) *and features that are additive over each sample, the dual optimization formulation (Eq.* (2)*) can be equivalently computed as:*

$$\max_\theta \Bigg\{ \min_{\mathbf{Q}\in\Delta} \max_{\mathbf{P}\in\Delta} \Bigg[\sum_{k,l\in[0,n]} \sum_j \tfrac{1}{g_j(k,l)} \Big\{ a_j[\mathbf{p}_k^1 \cdot \mathbf{q}_l^1] + b_j[\mathbf{p}_k^0 \cdot \mathbf{q}_l^0] + f_j(k,l) r_k s_l \Big\} - \langle \mathbf{Q}^\intercal \mathbf{1}, \Psi^\intercal \theta \rangle \Bigg] + \langle \mathbf{y}, \Psi^\intercal \theta \rangle \Bigg\}, \tag{9}$$

where Δ is the set of valid marginal probability matrices denoted as:

$$\Delta = \left\{ \mathbf{P} \;\middle|\; \begin{array}{ll} p_{i,k} \geq 0 & \forall i,k \in [1,n] \\ p_{i,k} \leq \frac{1}{k} \sum_j p_{j,k} & \forall i,k \in [1,n] \\ \sum_k \frac{1}{k} \sum_i p_{i,k} \leq 1 & \end{array} \right\}. \tag{10}$$

All of the terms in the objective: $\mathbf{p}_k^1$, $\mathbf{q}_l^1$, $\mathbf{p}_k^0$, $\mathbf{q}_l^0$, r_k, s_l, $\mathcal{P}(\mathbf{0})$, and $\mathcal{Q}(\mathbf{0})$ can be computed from $\mathbf{P}$ and $\mathbf{Q}$.

Using the construction above, we reduce the number of optimized variables in the inner minimax from exponential size to just quadratic size. Note that the objective in Eq. (9) remains bilinear over the optimized variables ($\mathbf{P}$ and $\mathbf{Q}$), as in the original formulation (Eq. (2)) that is bilinear over $\mathcal{P}(\hat{\mathbf{Y}})$ and $\mathcal{Q}(\check{\mathbf{Y}})$.

[3] The proof of this theorem and others in the paper are contained in Appendix A.

3.3 Optimization

One of the benefits of optimizing a loss metric using the adversarial prediction framework is that the resulting dual optimization (e.g., Eq. (2) and Eq. (9)) is convex (or concave in our case of optimizing performance metric) in θ, despite the original metric that we want to optimize is non-convex and non-continuous. Therefore, to achieve the global solution of θ, we can just use the standard gradient ascent algorithm. The sub-gradient of the objective with respect to theta is described in the following theorem.

Theorem 3. *Let $\mathcal{L}(\theta)$ be the objective of the maximization over θ in Eq. (9) and let $\mathbf{Q}^*$ be the solution of the inner minimization over $\mathbf{Q}$, then the sub-gradient of $-\mathcal{L}(\theta)$ with respect to θ includes:*

$$\partial_\theta - \mathcal{L}(\theta) \ni \Psi\left(\mathbf{Q}^{*\intercal}\mathbf{1} - \mathbf{y}\right). \tag{11}$$

To solve the inner minimax over $\mathbf{Q}$ and $\mathbf{P}$, we eliminate the inner-most optimization over $\mathbf{P}$ by introducing dual variables over some of the constraints on $\mathbf{P}$ and a slack variable that convert maximization into a set of constraints over $\mathbf{Q}$ and the slack variable. This results in a linear program optimization problem.

Theorem 4. *The inner minimization over $\mathbf{Q}$ in Eq. (9) can be solved as a linear program in the form of:*

$$\begin{aligned}&\min_{\mathbf{Q}\in\Delta;\boldsymbol{\alpha}\geq 0;v\geq 0} \; v + c(\mathbf{Q}) - \langle \mathbf{Q}, \Psi^\intercal\theta\mathbf{1}^\intercal\rangle \qquad (12)\\ &s.t.:\; v \geq \mathbf{Z}(\mathbf{Q})_{(i,k)} - \alpha_{i,k} + \tfrac{1}{k}\textstyle\sum_j \alpha_{j,k}, \quad \forall i,k \in [1,n],\end{aligned}$$

where $c(\mathbf{Q})$ is a linear function of $\mathbf{Q}$ and $\mathbf{Z}(\mathbf{Q})$ is a matrix-valued linear function of $\mathbf{Q}$, both of which are defined analytically by the form of the metric.[4]

3.4 Metric Constraints

In some machine learning settings, we may want to optimize a performance metric subject to constraints on other metrics. This occurs in the case where there are trade-offs between different performance metrics. For example, a machine learning system may want to optimize the precision of the prediction, subject to its recall is greater than some threshold. For these tasks, we write the adversarial prediction formulation as:

$$\begin{aligned}&\max_{\mathcal{P}(\check{\mathbf{Y}})}\min_{\mathcal{Q}(\hat{\mathbf{Y}})} \; \mathbb{E}_{\tilde{P}(\mathbf{X});\mathcal{P}(\hat{\mathbf{Y}});\mathcal{Q}(\check{\mathbf{Y}})}\left[\text{metric}^{(0)}(\hat{\mathbf{Y}},\check{\mathbf{Y}})\right] \qquad (13)\\ &\text{s.t.: } \mathbb{E}_{\tilde{P}(\mathbf{X});\mathcal{Q}(\check{\mathbf{Y}})}[\phi(\mathbf{X},\check{\mathbf{Y}})] = \mathbb{E}_{\tilde{P}(\mathbf{X},\mathbf{Y})}\left[\phi(\mathbf{X},\mathbf{Y})\right],\\ &\qquad \mathbb{E}_{\tilde{P}(\mathbf{X},\mathbf{Y});\mathcal{P}(\hat{\mathbf{Y}})}\left[\text{metric}^{(i)}(\hat{\mathbf{Y}},\mathbf{Y})\right] \geq \tau_i,\; \forall i \in [1,t],\end{aligned}$$

where t is the number of metric constraints. In this formulation, we constraint the predictor to choose a conditional distribution in which the expected values of the constraint metrics evaluated on empirical data are greater than some threshold τ.

As in Section 3.2, we use a marginalization technique to reduce the size of the optimization problem as stated in Theorem 5.

Theorem 5. *Let $\mathbf{P}$ and $\mathbf{Q}$ be the marginal predictor and adversary probability matrices respectively. Given a performance metric that follows the construction in Eq. (3), a set of constraints over metrics that also follows the construction in Eq. (3), and features that are additive over each sample, the dual optimization formulation of (Eq. (13)) can be computed as:*

$$\begin{aligned}\max_\theta\Bigg\{&\min_{\mathbf{Q}\in\Delta}\max_{\mathbf{P}\in\Delta\cap\Gamma}\Bigg[\sum_{k,l\in[0,n]}\sum_j \tfrac{1}{g_j^{(0)}(k,l)}\Big\{a_j^{(0)}[\mathbf{p}_k^1\cdot\mathbf{q}_l^1] \qquad (14)\\ &+b_j^{(0)}[\mathbf{p}_k^0\cdot\mathbf{q}_l^0]+f_j^{(0)}(k,l)r_k s_l\Big\} - \langle\mathbf{Q}^\intercal\mathbf{1},\Psi^\intercal\theta\rangle\Bigg] + \langle\mathbf{y},\Psi^\intercal\theta\rangle\Bigg\},\end{aligned}$$

where Δ is the set of marginal probability matrices defined in Eq (10), and Γ is the set of marginal probability matrices defined as:

$$\begin{aligned}\Gamma = \Bigg\{\mathbf{P}\Bigg|\; &\sum_{k\in[0,n]}\sum_j \tfrac{1}{g_j^{(i)}(k,l)}\Big\{a_j^{(i)}[\mathbf{p}_k^1\cdot\mathbf{y}] + b_j^{(i)}[\mathbf{p}_k^0\cdot(1-\mathbf{y})]\\ &+f_j^{(0)}(k,l)r_k\Big\} \geq \tau_i, \forall i\in[1,t]\Bigg\}, \text{where } l = \textstyle\sum_{i'} y_{i'}. \qquad (15)\end{aligned}$$

All of the terms in the objective: $\mathbf{p}_k^1$, $\mathbf{q}_l^1$, $\mathbf{p}_k^0$, $\mathbf{q}_l^0$, r_k, s_l, $\mathcal{P}(\mathbf{0})$, and $\mathcal{Q}(\mathbf{0})$, can be computed from $\mathbf{P}$ and $\mathbf{Q}$.

Note that the resulting optimization in the case where we have metric constraints (Eq. (14)) is relatively similar with the standard case (Eq. (9)). The only difference is the additional constraints over $\mathbf{P}$. Since the constraints in the set Γ are also just linear constraints over $\mathbf{P}$, we can also rewrite the inner minimization over $\mathbf{Q}$ in Eq. (14) as a linear program.

Theorem 6. *The inner minimization over $\mathbf{Q}$ in Eq. (14) can be solved as a linear program in the form of:*

$$\begin{aligned}&\min_{\mathbf{Q}\in\Delta;\boldsymbol{\alpha}\geq 0;\boldsymbol{\beta}\geq 0;v\geq 0} \; v + c(\mathbf{Q}) - \langle\mathbf{Q},\Psi^\intercal\theta\mathbf{1}^\intercal\rangle + \textstyle\sum_l(\mu_l - \tau_l)\\ &s.t.:\; v \geq \mathbf{Z}(\mathbf{Q})_{(i,k)} - \alpha_{i,k} + \tfrac{1}{k}\textstyle\sum_j\alpha_{j,k} + \sum_l \beta_l\,(\mathbf{B}^{(l)})_{(i,k)}\\ &\forall i,k\in[1,n], \qquad (16)\end{aligned}$$

where $c(\mathbf{Q})$ is a linear function of $\mathbf{Q}$ and $\mathbf{Z}(\mathbf{Q})$ is a matrix-valued linear function of $\mathbf{Q}$, both of which are defined analytically by the form of the metric; whereas μ_l is a constant and $\mathbf{B}^{(l)}$ is a matrix, both of which are defined analytically by the l-th metric constraint and the ground truth label.

3.5 Integration into Differentiable Learning

In this section, we aim to integrate our formulation into differentiable learning pipelines with a focus on

[4] Please see Appendix A for the details.

```
@metric Kappa
function define(::Type{Kappa}, C::ConfusionMatrix)
    pe = (C.ap * C.pp + C.an * C.pn) / C.all^2
    num = (C.tp + C.tn) / C.all - pe
    den = 1 - pe
    return num / den
end
kappa = Kappa()
special_case_positive!(kappa)
special_case_negative!(kappa)
```

Figure 2: Code example for Cohen's kappa score.

```
@metric PR
function define(::Type{PR}, C::ConfusionMatrix)
    return C.tp / C.pp
end
function constraint(::Type{PR}, C::ConfusionMatrix)
    return C.tp / C.ap >= 0.8
end
prec_rec = PR()
special_case_positive!(prec_rec)
cs_special_case_positive!(prec_rec, true)
```

Figure 3: Code example for precision metric with a constraint on recall metric.

those based upon neural network architectures. First, we note that even though we have reduced the number of variables in our formulation from exponential to quadratic size, it is still too big for most neural network learning tasks since the number of examples is usually big. Therefore, when optimizing the inner minimax over $\mathbf{Q}$ and $\mathbf{P}$, rather than optimizing over all samples, we perform optimization for every minibatch which limits the size of optimized variables into a relatively small quadratic size. We introduce non-linearity into our model by using the last layer of neural networks model as the features that we use to constraints the adversary's distribution in Eq. (1). Consequently, in the training process, we propagate back the gradient signal in Eq. (11) to the network.

To enable easy integration with machine learning pipelines, we develop a programming interface for writing a custom performance metric. This interface enables the user to write an arbitrary complex performance metric based on the entities in the confusion matrix. If the metric is valid according to our metric construction in Eq (3), we create an expression tree that stores all the operations in the metric. This expression tree is then used when computing the objective and constraints in Eq. (9) and Eq. (14) as well as the LP formulations in Eq. (12) and Eq. (16). We implement our method on top of Julia programming language (Bezanson et al., 2017) and its machine learning framework, FluxML (Innes et al., 2018). However, our method can be implemented in any other languages and frameworks. Figure 2 provides a code example for writing the definition of Cohen's kappa score metric. Note that our programming interface can handle a relatively complex performance metric. Figure 3 shows an example where we want to optimize precision, with a constraint that the recall has to be greater than 0.8. For more examples of the code for various performance metrics, we refer the reader to Appendix C.

3.6 Linear Program Solver and Runtime

As mentioned in Section 3.3, the inner minimization in the dual optimization of the adversarial prediction framework can be reformulated as a linear program (LP), which can be solved using any off-the-shelf LP solver such as Gurobi, Mosek, and Clp. The number of variables and constraints in the LP is $O(m^2)$, where m is the batch size. The worst-case complexity of solving a linear program is $O(n^3)$ using the interior point algorithm where n the number of variables. Therefore, the worst-case complexity of solving for the LP is $O(m^6)$ (solvers that exploit sparsity may reduce the runtime).

To reduce the runtime complexity of solving the resulting LP, we develop a customized solver using the alternating direction method of multipliers (ADMM) technique (Douglas and Rachford, 1956; Glowinski and Marroco, 1975; Boyd et al., 2011). This reduces the worst-case runtime complexity to just $O(m^3)$, where m is the batch size. In practice, for a batch size of 25, our ADMM-based solver takes roughly 10 - 30 milliseconds to converge in a desktop PC with an Intel Core i7 processor. While it is noticeably slower than the cross-entropy loss computation, it is still practical, since for reasonably sized networks, the loss function computation is usually dominated by the computation of the previous layers. We refer the reader to Appendix D for the detailed formulation of our custom solver.

3.7 Fisher Consistency Property

The behavior of a learning algorithm in an ideal setting (i.e., where the algorithm is given access to the true population distribution, and it is allowed to be optimized over the set of all measurable hypothesis functions), provides a useful theoretical validation. Fisher consistency requires that the prediction model yields the Bayes optimal decision boundary in this setting (Tewari and Bartlett, 2007; Liu, 2007) The Fisher consistency of the adversarial prediction framework has been established previously for decomposable metrics, bipartite matching, and graphical model (Fathony et al., 2018a,b,c). We establish the consistency of our approach in the following theorem.

Theorem 7. *Given a performance metric that follows the construction in Eq.* (3)*, the adversarial pre-*

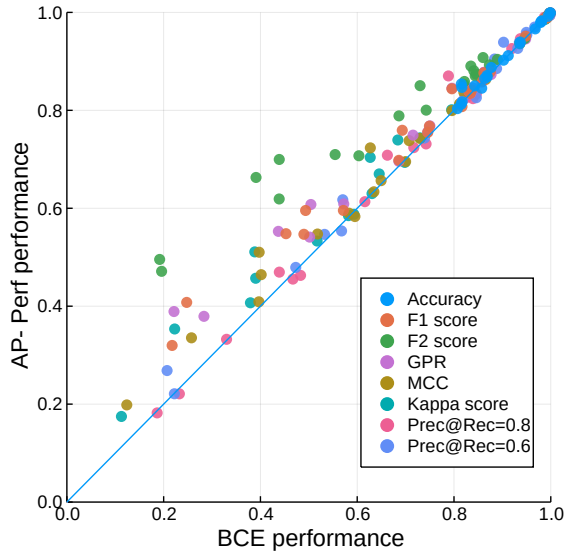

Figure 4: Comparison between BCE and AP-Perf.

diction formulation in Eq. (1) *is Fisher consistent if the algorithm is optimized over a set of functions that are additive over each sample and sum statistics, i.e.,* $h(\mathbf{x}, \mathbf{y}) = \sum_{i,k} \rho_{i,k}(\mathbf{x}_i, y_i, k)\mathbb{I}[\sum_i y_i = k]$*, provided that* $\rho_{i,j}$ *is allowed to be optimized over the set of all measurable functions on the individual input space* $(\mathbf{x}_i, y_i)$*.*

4 EXPERIMENTS

To evaluate our approach, we apply our formulation to classification tasks on 20 different tabular datasets from UCI repository (Dua and Graff, 2017) and benchmark datasets (Chu and Ghahramani, 2005), as well as image datasets from MNIST and Fashion MNIST. For the multiclass datasets, we transform them into binary classification tasks by selecting one or more classes as the positive label and the rest as the negative label. We construct a multi-layer perceptron (MLP) with two hidden layers for the tabular datasets and a convolutional neural network for the image datasets. We evaluate the prediction using 6 different metrics: accuracy, F1 score, F2 score, the geometric mean of precision and recall (GPR), Matthews correlation coefficient (MCC), and Cohen's kappa score. We also evaluate the prediction using metric constraints, specifically, we train our method to optimize precision given that the recall is greater than certain thresholds. We select two different thresholds for the recall, 0.8 and 0.6. We then measure the prediction using precision at recall equal to the given thresholds.

We compare our method with the standard neural networks training that optimizes the binary cross-entropy (BCE) on the 22 datasets. In our experiment, we train our methods separately for each performance metric that we want to optimize, whereas for the BCE networks, we only train the networks once using the cross-entropy objective. We then measure the performance of the prediction using 8 metrics that we have selected. For both methods, we perform a cross-validation using validation set to select the best L2 regularization among $\lambda = \{0, 0.001, 0.01, 0.1\}$. In each dataset, we run the training procedure for 100 epochs. After the training session finished, we compute the value of the metric for prediction in the testing dataset. For both methods, we select the predictive models that achieve the best metric in the validation set. We refer the reader to Appendix B for the details about the datasets and experiment setup. The AP-Perf framework code is available at `https://github.com/ rizalzaf/ AdversarialPrediction.jl`, whereas the experiment is available at `https://github.com/rizalzaf/AP-examples`.

Figure 4 shows a scatter plot of the comparison between our method and the BCE on the 22 datasets. The x-axis in the plot denotes the BCE performance whereas the y-axis is the AP-Perf performance. The blue line in the plot denotes the case where the performance for both methods is equal. Points above the line indicate that AP-Perf outperforms the BCE on the particular dataset and evaluation metric. As we can see from the figure, almost all of the points in the scatter plot lie on or above the blue line. This shows the benefit of our methods in optimizing the performance metrics as opposed to training the network using the cross-entropy objective. From the plot, we can infer that our method provides more benefit for the "relatively hard problems", i.e., the tasks where the BCE produces good accuracy but low to moderate performance in other metrics. We can also see that the AP-Perf consistently provides the best improvement over the BCE on the F2 score metric. This can be explained by the fact that the F2 score is the only imbalance metric from the list, i.e., it emphasizes some parts of the metric (in F2-score, recall is two times as important as precision). Since the BCE optimizes a proxy to a balanced metric (accuracy), it suffers more in the case where an imbalance metric is used for evaluation.

5 CONCLUSION

We developed a technique and programming interface that enable practitioners to integrate custom non-decomposable metric into differentiable learning. Our methods support a vast range of commonly used performance metrics. The list of metrics that our approach support is, however, far from exhaustive. The most noticeable missing metric is the area-based metric (e.g., AUC-ROC), which cannot be directly computed from the value of the entities in the confusion matrix, and ranking-based metrics (e.g., precision@k and MAP). Our future works aim to close these gaps in the metric that we do not support.

References

Kaiser Asif, Wei Xing, Sima Behpour, and Brian D. Ziebart. Adversarial cost-sensitive classification. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 92–101, 2015.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. URL https://doi.org/10.1137/141000671.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Róbert Busa-Fekete, Balázs Szörényi, Krzysztof Dembczynski, and Eyke Hüllermeier. Online f-measure optimization. In *Advances in Neural Information Processing Systems*, pages 595–603, 2015.

Wei Chu and Zoubin Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6(Jul):1019–1041, 2005.

Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. *Advances in Neural Information Processing Systems*, 16(16):313–320, 2004.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240. ACM, 2006.

Krzysztof J Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. An exact algorithm for f-measure maximization. In *Advances in Neural Information Processing Systems*, pages 1404–1412, 2011.

Jim Douglas and Henry H Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, 82(2):421–439, 1956.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Elad Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Ryan Rifkin, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial Intelligence and Statistics*, pages 832–840, 2017.

Rizal Fathony, Anqi Liu, Kaiser Asif, and Brian Ziebart. Adversarial multiclass classification: A risk minimization perspective. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 559–567, 2016.

Rizal Fathony, Mohammad Ali Bashiri, and Brian Ziebart. Adversarial surrogate losses for ordinal regression. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 563–573, 2017.

Rizal Fathony, Kaiser Asif, Anqi Liu, Mohammad Ali Bashiri, Wei Xing, Sima Behpour, Xinhua Zhang, and Brian D Ziebart. Consistent robust adversarial prediction for general multiclass classification. *arXiv preprint arXiv:1812.07526*, 2018a.

Rizal Fathony, Sima Behpour, Xinhua Zhang, and Brian Ziebart. Efficient and consistent adversarial bipartite matching. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2017)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1457–1466, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR.

Rizal Fathony, Ashkan Rezaei, Mohammad Ali Bashiri, Xinhua Zhang, and Brian Ziebart. Distributionally robust graphical models. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 8353–8364, 2018c.

Roland Glowinski and A Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9 (R2):41–76, 1975.

Tamir Hazan, Joseph Keshet, and David A McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2010.

Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal Singh, and Viral Shah. Fashionable modelling with Flux. *NeurIPS Workshop on Systems for Machine Learning (MLSys)*, 2018.

Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, 2005.

Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Online and stochastic gradient methods for non-decomposable loss functions. In *Advances in Neural Information Processing Systems*, pages 694–702, 2014.

Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In *International Conference on Machine Learning*, pages 189–198, 2015.

Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent binary classification with generalized

performance metrics. In *Advances in Neural Information Processing Systems*, pages 2744–2752, 2014.

Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize f1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–239. Springer, 2014.

Yufeng Liu. Fisher consistency of multicategory support vector machines. In *International Conference on Artificial Intelligence and Statistics*, pages 291–298, 2007.

JJ Moreau. Decomposition orthogonale d'un espace hilbertien selon deux cones mutuellement polaires. *CR Acad. Sci. Paris*, 225:238–240, 1962.

Harikrishna Narasimhan and Shivani Agarwal. A structural svm based approach for optimizing partial auc. In *International Conference on Machine Learning*, pages 516–524, 2013a.

Harikrishna Narasimhan and Shivani Agarwal. Svm pauc tight: a new support vector method for optimizing partial auc based on a tight convex upper bound. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 167–175. ACM, 2013b.

Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Advances in Neural Information Processing Systems*, pages 1493–1501, 2014.

Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *International Conference on Machine Learning*, pages 199–208, 2015.

Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing F-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, pages 2123–2131, 2014.

David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Technical Report, Flinders University*, 2011.

Amartya Sanyal, Pawan Kumar, Purushottam Kar, Sanjay Chawla, and Fabrizio Sebastiani. Optimizing non-decomposable measures with deep networks. *Machine Learning*, 107(8-10):1597–1620, 2018.

Zhan Shi, Xinhua Zhang, and Yaoliang Yu. Bregman divergence for stochastic variance reduction: Saddle-point and adversarial prediction. In *Advances in Neural Information Processing Systems*, pages 6033–6043, 2017.

Maurice Sion. On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176, 1958.

Yang Song, Alexander Schwing, Raquel Urtasun, et al. Training deep neural networks via direct loss minimization. In *International Conference on Machine Learning*, pages 2169–2177, 2016.

Ambuj Tewari and Peter L Bartlett. On the consistency of multiclass classification methods. *The Journal of Machine Learning Research*, 8:1007–1025, 2007.

Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, pages 831–838, 1992.

John Von Neumann and Oskar Morgenstern. Theory of games and economic behavior. *Bulletin of the American Mathematical Society*, 51(7):498–504, 1945.

Hong Wang, Wei Xing, Kaiser Asif, and Brian Ziebart. Adversarial prediction games for multivariate losses. In *Advances in Neural Information Processing Systems*, pages 2710–2718, 2015.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–278. ACM, 2007.