Online Optimization Methods for the Quantification Problem*

Purushottam Kar
IIT Kanpur, India
purushot@cse.iitk.ac.in

Shuai Li†
University of Insubria, Italy
shuaili.sli@gmail.com

Harikrishna Narasimhan
Harvard University, USA
hnarasimhan@g.harvard.edu

Sanjay Chawla‡
QCRI-HBKU, Qatar
schawla@qf.org.qa

Fabrizio Sebastiani§
QCRI-HBKU, Qatar
fsebastiani@qf.org.qa

Abstract

The estimation of class prevalence, i.e., the fraction of a population that belongs to a certain class, is a very useful tool in data analytics and learning, and finds applications in many domains such as sentiment analysis, epidemiology, etc. For example, in sentiment analysis, the objective is often not to estimate whether a specific text conveys a positive or a negative sentiment, but rather estimate the overall distribution of positive and negative sentiments during an event window. A popular way of performing the above task, often dubbed *quantification*, is to use supervised learning to train a prevalence estimator from labeled data.

Contemporary literature cites several performance measures used to measure the success of such prevalence estimators. In this paper we propose the first online stochastic algorithms for *directly* optimizing these quantification-specific performance measures. We also provide algorithms that optimize *hybrid* performance measures that seek to balance quantification and classification performance. Our algorithms present a significant advancement in the theory of multivariate optimization and we show, by a rigorous theoretical analysis, that they exhibit optimal convergence. We also report extensive experiments on benchmark and real data sets which demonstrate that our methods significantly outperform existing optimization techniques used for these performance measures.

1 Introduction

Quantification [11] is defined as the task of estimating the prevalence (i.e., relative frequency) of the classes of interest in an unlabeled set, given a training set of items labeled according to the same classes. Quantification finds its natural application in contexts characterized by *distribution drift*, i.e., contexts where the training data may not exhibit the same class prevalence pattern as the test data. This phenomenon may be due to different reasons, including the inherent non-stationary character of the context, or class bias that affects the selection of the training data.

*A short version of this manuscript will appear in the proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2016.

†This work was done while Shuai Li was a research associate at QCRI-HBKU.

‡Sanjay Chawla is on leave from University of Sydney.

§Fabrizio Sebastiani is on leave from Consiglio Nazionale delle Ricerche, Italy.

A naïve way to tackle quantification is via the "classify and count" (CC) approach, i.e., to classify each unlabeled item independently and compute the fraction of the unlabeled items that have been attributed to each class. However, a good classifier does not necessarily lead to a good quantifier: assuming the binary case, even if the sum (FP + FN) of the false positives and false negatives is comparatively small, bad quantification accuracy might result if FP and FN are significantly different (since perfect quantification coincides with the case FP = FN). This has led researchers to study quantification as a task in its own right, rather than as a byproduct of classification.

The fact that quantification is not just classification in disguise can also be seen by the fact that evaluation measures different from those for classification (e.g., F_1, AUC) need to be employed. Quantification actually amounts to computing how well an estimated class distribution $\hat{p}$ fits an actual class distribution p (where for any class $c \in \mathcal{C}$, $p(c)$ and $\hat{p}(c)$ respectively denote its true and estimated prevalence); as such, the natural way to evaluate the quality of this fit is via a function from the class of *f-divergences* [7], and a natural choice from this class (if only for the fact that it is the best known f-divergence) is the *Kullback-Leibler Divergence* (KLD), defined as

$$\text{KLD}(p, \hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log \frac{p(c)}{\hat{p}(c)} \tag{1}$$

Indeed, KLD is the most frequently used measure for evaluating quantification (see e.g., [3, 10, 11, 12]). Note that KLD is non-decomposable, i.e., the error we make by estimating p via $\hat{p}$ cannot be broken down into item-level errors. This is not just a feature of KLD, but an inherent feature of *any* measure for evaluating quantification. In fact, how the error made on a given unlabeled item impacts the overall quantification error depends on how the other items have been classified[1]; e.g., if FP $>$ FN for the other unlabeled items, then generating an additional false negative is actually *beneficial* to the overall quantification accuracy, be it measured via KLD or via any other function.

The fact that KLD is the measure of choice for quantification and that it is non-decomposable, has lead to the use of structured output learners, such as SVM^{perf} [17], that allow a direct optimization of non-decomposable functions; the approach of Esuli and Sebastiani [9, 10] is indeed based on optimizing KLD using SVM^{perf}. However, that minimizing KLD (or $|\text{FP} - \text{FN}|$, or any "pure" quantification measure) should be the only objective for quantification regardless of the value of FP + FN (or any other classification measure), is fairly paradoxical. Some authors [3, 24] have observed that this might lead to the generation of unreliable quantifiers (i.e., systems with good quantification accuracy but bad or very bad classification accuracy), and have, as a result, championed the idea of optimizing "multi-objective" measures that combine quantification accuracy with classification accuracy. Using a decision-tree-like approach, [24] minimizes $|\text{FP}^2 - \text{FN}^2|$, which is the product of $|\text{FN} - \text{FP}|$, a measure of quantification error, and $(\text{FN} + \text{FP})$, a measure of classification error; [3] also optimizes (using SVM^{perf}) a measure that combines quantification and classification accuracy.

While SVM^{perf} does provide a recipe for optimizing general performance measures, it has serious limitations. SVM^{perf} is not designed to directly handle applications where large streaming data sets are the norm. SVM^{perf} also does not scale well to multi-class settings, and the time required by the method is exponential in the number of classes.

In this paper we develop stochastic methods for optimizing a large family of popular quantification performance measures. Our methods can effortlessly work with streaming data and scale to very large datasets, offering training times up to an order of magnitude faster than other approaches such as SVM^{perf}.

2 Related Work

Quantification methods. The quantification methods that have been proposed over the years can be broadly classified into two classes, namely aggregative and non-aggregative methods. While *aggregative* approaches perform quantification by first classifying individual items as an intermediate step, *non-aggregative* approaches do not require this step, and estimate class prevalences holistically. Most methods, such as those of [3, 4, 10, 11, 24], fall in the former class, while the latter class has few representatives [14, 22].

[1] For the sake of simplicity, we assume here that quantification is to be tackled in an *aggregative* way, i.e., the classification of individual items is a necessary intermediate step for the estimation of class prevalences. Note however that this is not necessary; non-aggregative approaches to quantification may be found in [14, 22].

Within the class of aggregative methods, a further distinction can be made between methods, such as those of [4, 11], that first use general-purpose learning algorithms and then post-process their prevalence estimates to account for their estimation biases, and methods (which we have already hinted at in Section 1) that instead use learning algorithms explicitly devised for quantification [3, 10, 24]. In this paper we focus the latter class of methods.

Applications of quantification. From an application perspective, quantification is especially useful in fields (such as social science, political science, market research, and epidemiology) which are inherently interested in aggregate data, and care little about individual cases. Aside from applications in these fields [16, 22], quantification has also been used in contexts as diverse as natural language processing [6], resource allocation [11], tweet sentiment analysis [12], and the veterinary sciences [14]. Quantification has independently been studied within statistics [16, 22], machine learning [2, 8, 29], and data mining [10, 11]. Unsurprisingly, given this varied literature, quantification also goes under different names, such as *counting* [23], *class probability re-estimation* [1], *class prior estimation* [6], and *learning of class balance* [8].

In some applications of quantification, the estimation of class prevalences is not an end in itself, but is rather used to improve the accuracy of other tasks such as classification. For instance, Balikas et al. [2] use quantification for model selection in supervised learning, by tuning hyperparameters that yield the best quantification accuracy on validation data; this allows hyperparameter tuning to be performed without incurring the costs inherent to k-fold cross-validation. Saerens et al. [29], followed by other authors [1, 32, 34], apply quantification to customize a trained classifier to the class prevalence exhibited in the test set, with the goal of improving classification accuracy on unlabeled data exhibiting a class distribution different from that of the training set. The work of Chan and Ng [6] may be seen as a direct application of this notion, as they use quantification to tune a word sense disambiguator to the estimated sense priors of the test set. Their work can also be seen as an instance of *transfer learning* (see e.g., [26]), since their goal is to adapt a word sense disambiguation algorithm to a domain different from the one the algorithm was trained upon.

Stochastic optimization. As discussed in Section 1, our goal in this paper is to perform quantification by directly optimizing, in an online stochastic setting, specific performance measures for the quantification problem. While recent advances have seen much progress in efficient methods for online learning and optimization in full information and bandit settings [5, 13, 15, 30], these works frequently assume that the optimization objective, or the notion of regret being considered is decomposable and can be written as a sum or expectation of losses or penalties on individual data points. However, performance measures for quantification have a multivariate and complex structure, and do not have this form.

There has been some recent progress [20, 25] towards developing stochastic optimization methods for such non-decomposable measures. However, these approaches do not satisfy the needs of our problem. The work of Kar et al. [20] addresses the problem of optimizing structured SVM^{perf}-style objectives in a streaming fashion, but requires the maintenance of large buffers and, as a result, offers poor convergence. The work of Narasimhan et al. [25] presents online stochastic methods for optimizing performance measures that are concave or pseudo-linear in the canonical confusion matrix of the predictor. However, their method requires the computation of gradients of the Fenchel dual of the performance measures, which is difficult for the quantification performance measures that we study, that have a nested structure. Our methods extend the work of [25] and provide convenient routines for optimizing the more complex performance measures used for evaluating quantification.

3 Problem Setting

For the sake of simplicity, in this paper we will restrict our analysis to binary classification problems and linear models. We will denote the space of feature vectors by $\mathcal{X} \subset \mathbb{R}^d$ and the label set by $\mathcal{Y} = \{-1, +1\}$. We shall assume that data points are generated according to some fixed but unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. We will denote the proportion of positives in the population by $p := \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[y = +1]$. Our algorithms, at training time, will receive a set of T training points sampled from $\mathcal{D}$, which we will denote by $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$.

As mentioned above, we will present our algorithms and analyses for learning a linear model over $\mathcal{X}$. We will denote the model space by $\mathcal{W} \subseteq \mathbb{R}^d$ and let $R_{\mathcal{X}}$ and $R_{\mathcal{W}}$ denote the radii of domain $\mathcal{X}$ and model space $\mathcal{W}$, respectively. However, we note that our algorithms and analyses can be extended to learning non-linear models by use of kernels, as well as to multi-class quantification problems. However, we postpone a discussion of these extensions to an expanded version of this paper.

Our focus in this work shall be the optimization of quantification-specific performance measures in online stochastic settings. We will concentrate on performance measures that can be represented as functions of the confusion matrix of the classifier. In the binary setting, the confusion matrix can be completely described in terms of the true positive rate (TPR) and the true negative rate (TNR) of the classifier. However, initially we will develop algorithms that use *reward functions* as surrogates of the TPR and TNR values. This is done to ease algorithm design and analysis, since the TPR and TNR values are count-based and form non-concave and non-differentiable estimators. The surrogates we will use will be concave and almost-everywhere differentiable. More formally, we will use a reward function r that assigns a *reward* $r(\hat{y}, y)$ to a prediction $\hat{y} \in \mathbb{R}$ for a data point, when the true label for that data point is $y \in \mathcal{Y}$. Given a reward function r, a model $\mathbf{w} \in \mathcal{W}$, and a data point $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, we will use

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

to calculate rewards on positive and negative points. The average or expected value of these rewards will be treated as surrogates of TPR and TNR respectively. Note that since $\mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r^+(\mathbf{w}; \mathbf{x}, y)]\!] = \mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r(\mathbf{w}^\top \mathbf{x}, y) | y = 1]\!]$, setting r to $r^{0\text{-}1}(\hat{y}, y) = \mathbb{I}[y \cdot \hat{y} > 0]$, i.e. the classification accuracy function, yields $\mathop{\mathbb{E}}_{(\mathbf{x},y)} [\![r^+(\mathbf{w}; \mathbf{x}, y)]\!] = \text{TPR}(\mathbf{w})$. Here $\mathbb{I}[\cdot]$ denotes the indicator function. For the 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 function r is concave, L_r-Lipschitz, and takes values in a bounded range $[-B_r, B_r]$.

Examples of Surrogate Reward Functions Some examples of reward functions that are surrogates for the classification accuracy indicator function $\mathbb{I}[y\hat{y} > 0]$ are the inverted hinge loss function

$$r_{\text{hinge}}(\hat{y}, y) = \max\{1, y \cdot \hat{y}\}$$

and the inverted logistic regression function

$$r_{\text{logit}}(\hat{y}, y) = 1 - \ln(1 + \exp(-y \cdot \hat{y}))$$

We will also experiment with non-surrogate (dubbed NS) versions of our algorithms which use TPR and TNR values directly. These will be discussed in Section 5.

3.1 Performance Measures

The task of quantification requires estimating the distribution of unlabeled items across a set $\mathcal{C}$ of available classes, with $|\mathcal{C}| = 2$ in the binary setting. In our work we will target quantification performance measures as well as "hybrid" classification-quantification performance measures. We discuss them in turn.

KLD: Kullback-Leibler Divergence In recent years this performance measure has become a standard in the quantification literature, in the evaluation of both binary and multiclass quantification [3, 10, 12]. We redefine KLD below for convenience.

$$\text{KLD}(p, \hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log \frac{p(c)}{\hat{p}(c)} \tag{2}$$

For distributions p, p' over $\mathcal{C}$, the values $\text{KLD}(p, p')$ can range between 0 (perfect quantification) and $+\infty$.[2]. Note that since KLD is a distance function and all our algorithms will be driven by reward maximization, for uniformity we will, instead of trying to minimize KLD, try to maximize $-$KLD; we will call this latter NegKLD.

NSS: Normalized Squared Score This measure of quantification accuracy was introduced in [3], and is defined as $\text{NSS} = 1 - (\frac{\text{FN}-\text{FP}}{\max\{p,(1-p)\}|\mathcal{S}|})^2$. Ignoring normalization constants, this performance measure attempts to reduce $|\text{FN} - \text{FP}|$, a direct measure of quantification error.

We recall from Section 1 that several works have advocated the use of hybrid, "multi-objective" performance measures, that try to balance quantification and classification performance. These measures typically take a quantification performance measure such as KLD or NSS, and combine it with a classification performance measure. Typically, a classification performance measure that is sensitive to class imbalance [25] is chosen, such as Balanced Accuracy $\text{BA} = \frac{1}{2}(\text{TPR} + \text{TNR})$ [3], F-measure, or G-mean [25]. Two such hybrid performance measures that are discussed in literature are presented below.

CQB: Classification-Quantification Balancing The work of [24] introduced this performance measure in an attempt to compromise between classification and quantification accuracy. As discussed in Section 1, this performance measure is defined as

$$\text{CQB} = |\text{FP}^2 - \text{FN}^2| = |\text{FP} - \text{FN}| \cdot (\text{FP} + \text{FN}),$$

i.e. a product of $|\text{FN} - \text{FP}|$, a measure of quantification error, and $(\text{FN} + \text{FP})$, a measure of classification error.

QMeasure The work of Barranquero et al. [3] introduced a generic scheme for constructing hybrid performance measures, using the so-called Q-measure defined as

$$Q_\beta = (1 + \beta^2) \cdot \frac{\mathcal{P}_{\text{class}} \cdot \mathcal{P}_{\text{quant}}}{\beta^2 \mathcal{P}_{\text{class}} + \mathcal{P}_{\text{quant}}}, \tag{3}$$

that is, a weighted combination of a measure of classification accuracy $\mathcal{P}_{\text{class}}$ and a measure of quantification accuracy $\mathcal{P}_{\text{quant}}$. For the sake of simplicity, in our experiments we will adopt $\text{BA} = \frac{1}{2}(\text{TPR} + \text{TNR})$ as our $\mathcal{P}_{\text{class}}$ and NSS as our $\mathcal{P}_{\text{quant}}$. However, we stress that our methods can be suitably adapted to work with other choices of $\mathcal{P}_{\text{class}}$ and $\mathcal{P}_{\text{quant}}$.

We also introduce three new hybrid performance measures in this paper as a way of testing our optimization algorithms. We define these below and refer the reader to Tables 1 and 2 for details.

BAKLD This hybrid performance measure takes a weighted average of BA and NegKLD; i.e. $\text{BAKLD} = C \cdot \text{BA} + (1 - C) \cdot (-\text{KLD})$. This performance measure gives the user a strong handle on how much emphasis should be placed on quantification and how much on classification performance. We will use BAKLD in our experiments to show that our methods offer an attractive tradeoff between the two.

[2]KLD is not a particularly well-behaved performance measure, since it is capable of taking unbounded values within the compact domain of the unit simplex. This poses a problem for optimization algorithms from the point of view of convergence, as well as numerical stability. To solve this problem, while computing KLD for two distributions p and $\hat{p}$, we can perform an *additive smoothing* of both $p(c)$ and $\hat{p}(c)$ by computing

$$p_s(c) = \frac{\epsilon + p(c)}{\epsilon|\mathcal{C}| + \sum_{c \in \mathcal{C}} p(c)}$$

where $p_s(c)$ denotes the smoothed version of $p(c)$. The denominator here is just a normalizing factor. The quantity $\epsilon = \frac{1}{2|\mathcal{S}|}$ is often used as a smoothing factor, and is the one we adopt here. The smoothed versions of $p(c)$ and $\hat{p}(c)$ are then used in place of the non-smoothed versions in Equation 1. We can show that, as a result, KLD is always bounded by $\text{KLD}(p_s, \hat{p}_s) \leq \mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ However, we note that the smoothed KLD still returns a value of 0 when p and $\hat{p}$ are identical distributions.

We now define two hybrid performance measures that are constructed by taking the *ratio* of a classification and a quantification performance measures. The aim of this exercise is to obtain performance measures that mimic the F-measure, which is also a pseudolinear performance measure [25]. The ability of our methods to directly optimize such complex performance measures will be indicative of their utility in terms of the freedom they allow the user to design objectives in a data- and task-specific manner.

CQReward and BKReward These hybrid performance measures are defined as $\text{CQReward} = \frac{\text{BA}}{2-\text{NSS}}$ and $\text{BKReward} = \frac{\text{BA}}{1+\text{KLD}}$. Notice that both performance measures are optimized when the numerator i.e. BA is large, and the denominator is small which translates to NSS being large for CQReward and KLD being small for BKReward. Clearly, both performance measures encourage good performance with respect to both classification and quantification and penalize a predictor which either neglects quantification to get better classification performance, or the other way round.

The past section has seen us introduce a wide variety of quantification and hybrid performance measures. Of these, the NegKLD, NSS, and Q-measure were already prevalent in quantification literature and we introduced BAKLD, CQReward and BKReward. As discussed before, the aim of exploring such a large variety of performance measures is to both demonstrate the utility of our methods with respect to the quantification problem, and present newer ways of designing hybrid performance measures that give the user more expressivity in tailoring the performance measure to the task at hand.

We also note that these performance measures have extremely diverse and complex structures. We can show that NegKLD, Q-measure, and BAKLD are nested concave functions, more specifically, concave functions of functions that are themselves concave in the confusion matrix of the predictor. On the other hand, CQReward and BKReward turn out to be pseudo-concave functions of the confusion matrix. Thus, we are working with two very different families of performance measures here, each of which has different properties and requires different optimization techniques.

In the following section, we introduce two novel methods to optimize these two families of performance measures.

4 Stochastic Optimization Methods for Quantification

The previous discussion in Sections 1 and 2 clarifies two aspects of efforts in the quantification literature. Firstly, specific performance measures have been developed and adopted for evaluating quantification performance including KLD, NSS, Q-measure etc. Secondly, algorithms that *directly* optimize these performance measures are desirable, as is evidenced by recent works [3, 9, 10, 24].

The works mentioned above make use of tools from optimization literature to learn linear (e.g. [10]) and non-linear (e.g. [24]) models to perform quantification. The state of the art efforts in this direction have adopted the structural SVM approach for optimizing these performance measures with great success [3, 10]. However, this approach comes with severe drawbacks.

The structural SVM [17], although a significant tool that allows optimization of arbitrary performance measures, suffers from two key drawbacks. Firstly, the structural SVM surrogate is not necessarily a tight surrogate for all performance measures, something that has been demonstrated in past literature [21, 25], which can lead to poor training. But more importantly, optimizing the structural SVM surrogate requires the use of expensive cutting plane methods which are known to scale poorly with the amount of training data, as well as are unable to handle streaming data.

To alleviate these problems, we propose stochastic optimization algorithms that *directly* optimize a large family of quantification performance measures. Our methods come with sound theoretical convergence guarantees, are able to operate with streaming data sets and, as our experiments will demonstrate, offer much faster and accurate quantification performance on a variety of data sets.

Our optimization techniques introduce crucial advancements in the field of stochastic optimization of *multivariate performance measures* and address the two families of performance measures discussed while concluding Section 3 – 1) nested concave performance measures and 2) pseudo-concave performance measures. We describe these in turn below.

Table 1: A list of nested concave performance measures and their canonical expressions in terms of the confusion matrix $\Psi(P, N)$ where P and N denote the TPR, TNR values and p and n denote the proportion of positives and negatives in the population. The 4th, 6th and 8th columns give the closed form updates used in steps 15-17 in Algorithm 1.

Name	Expression	$\Psi(x, y)$	$\boldsymbol{\gamma}(\mathbf{q})$	$\zeta_1(P, N)$	$\boldsymbol{\alpha}(\mathbf{r})$	$\zeta_2(P, N)$	$\boldsymbol{\beta}(\mathbf{r})$
NegKLD [3, 10]	$-\text{KLD}(p,\hat{p})$	$p{\cdot}x+n{\cdot}y$	(p,n)	$\log(pP+n(1-N))$	$\left(\frac{1}{r_1},\frac{1}{r_2}\right)$	$\log(nN+p(1-P))$	$\left(\frac{1}{r_1},\frac{1}{r_2}\right)$
QMeasure$_\beta$[3]	$\frac{(1+\beta^2)\cdot\text{BA}\cdot\text{NSS}}{\beta^2\cdot\text{BA}+\text{NSS}}$	$\frac{(1+\beta^2)\cdot x\cdot y}{\beta^2\cdot x+y}$	$(1+\beta^2)\cdot\left(z^2,\frac{(1-z)^2}{\beta^2}\right)$ $z=\frac{q_2}{\beta^2 q_1+q_2}$	$\frac{P+N}{2}$	$(\frac{1}{2},\frac{1}{2})$	$1-(p(1-P)-n(1-N))^2$	$\frac{2(z,-z)}{z=r_2-r_1}$
BAKLD	$C{\cdot}\text{BA}+(1-C){\cdot}(-\text{KLD})$	$C{\cdot}x+(1-C){\cdot}y$	$(C,1-C)$	$\frac{P+N}{2}$	$(\frac{1}{2},\frac{1}{2})$	$-\text{KLD}(P,N)$	see above

4.1 Nested Concave Performance Measures

The first class of performance measures that we deal with are concave combinations of concave performance measures. More formally, given three concave functions $\Psi, \zeta_1, \zeta_2 : \mathbb{R}^2 \to \mathbb{R}$, we can define a performance measure

$$\mathcal{P}_{(\Psi,\zeta_1,\zeta_2)}(\mathbf{w}) = \Psi(\zeta_1(\mathbf{w}), \zeta_2(\mathbf{w})),$$

where we have

$$\zeta_1(\mathbf{w}) = \zeta_1(P(\mathbf{w}), N(\mathbf{w}))$$
$$\zeta_2(\mathbf{w}) = \zeta_2(P(\mathbf{w}), N(\mathbf{w})),$$

where $P(\mathbf{w})$ and $N(\mathbf{w})$ can respectively denote, either the TPR and TNR values or surrogate reward functions therefor. Examples of such performance measures include the negative KLD performance measure and the QMeasure which are described in Section 3.1. Table 1 describes these performance measures in canonical form i.e. their expressions in terms of TPR and TNR values.

Before describing our algorithm for nested concave measures, we recall the notion of concave Fenchel conjugate of concave functions. For any concave function $f : \mathbb{R}^2 \to \mathbb{R}$ and any $(u, v) \in \mathbb{R}^2$, the (concave) Fenchel conjugate of f is defined as

$$f^*(u, v) = \inf_{(x,y)\in\mathbb{R}^2} \{ux + vy - f(x, y)\}.$$

Clearly, f^* is concave. Moreover, it follows from the concavity of f that for any $(x, y) \in \mathbb{R}^2$,

$$f(x, y) = \inf_{(u,v)\in\mathbb{R}^2} \{xu + yv - f^*(u, v)\}.$$

Below we state the properties of strong concavity and smoothness. These will be crucial in our convergence analysis.

Definition 1 (Strong Concavity and Smoothness)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be α-strongly concave and γ-smooth if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have*

$$-\frac{\gamma}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 \le f(\mathbf{x}) - f(\mathbf{y}) - \langle\nabla f(\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle \le -\frac{\alpha}{2}\|\mathbf{x}-\mathbf{y}\|_2^2.$$

We will assume that the functions Ψ, ζ_1, and ζ_2 defining our performance measures are γ-smooth for some constant $\gamma > 0$. This is true of all functions, save the $\log$ function which is used in the definition of the KLD quantification measure. However, if we carry out the smoothing step pointed out in Section 3.1 with some $\epsilon > 0$, then it can be shown that the KLD function does become $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$-smooth. An important property of smooth functions, that would be crucial in our analyses, is a close relationship between smooth and strongly convex functions

Theorem 2 ([33])**.** *A closed, concave function f is β smooth iff its (concave) Fenchel conjugate f^* is $\frac{1}{\beta}$-strongly concave.*

Algorithm 1 NEMSIS: NEsted priMal-dual StochastIc updateS

Require: Outer wrapper function Ψ, inner performance measures ζ_1, ζ_2, step sizes η_t, feasible sets $\mathcal{W}, \mathcal{A}$
Ensure: Classifier $\mathbf{w} \in \mathcal{W}$

1: $\mathbf{w}_0 \leftarrow \mathbf{0}, t \leftarrow 0, \{\mathbf{r}_0, \mathbf{q}_0, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0\} \leftarrow (0, 0)$
2: **while** data stream has points **do**
3: Receive data point $(\mathbf{x}_t, y_t)$
4: // `Perform primal ascent`
5: **if** $y_t > 0$ **then**
6: $\mathbf{w}_{t+1} \leftarrow \Pi_{\mathcal{W}} \left(\mathbf{w}_t + \eta_t (\gamma_{t,1} \alpha_{t,1} + \gamma_{t,2} \beta_{t,1}) \nabla_{\mathbf{w}} r^+(\mathbf{w}_t; \mathbf{x}_t, y_t) \right)$
7: $\mathbf{q}_{t+1} \leftarrow t \cdot \mathbf{q}_t + (\alpha_{t,1}, \beta_{t,1}) \cdot r^+(\mathbf{w}_t; \mathbf{x}_t, y_t)$
8: **else**
9: $\mathbf{w}_{t+1} \leftarrow \Pi_{\mathcal{W}} \left(\mathbf{w}_t + \eta_t (\gamma_{t,1} \alpha_{t,2} + \gamma_{t,2} \beta_{t,2}) \nabla_{\mathbf{w}} r^-(\mathbf{w}_t; \mathbf{x}_t, y_t) \right)$
10: $\mathbf{q}_{t+1} \leftarrow t \cdot \mathbf{q}_t + (\alpha_{t,2}, \beta_{t,2}) \cdot r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)$
11: **end if**
12: $\mathbf{r}_{t+1} \leftarrow (t+1)^{-1} \left(t \cdot \mathbf{r}_t + (r^+(\mathbf{w}_t; \mathbf{x}_t, y_t), r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)) \right)$
13: $\mathbf{q}_{t+1} \leftarrow (t+1)^{-1} \left(\mathbf{q}_{t+1} - (\zeta_1^*(\boldsymbol{\alpha}_t), \zeta_2^*(\boldsymbol{\beta}_t)) \right)$
14: // `Perform dual updates`
15: $\boldsymbol{\alpha}_{t+1} = \arg\min_{\boldsymbol{\alpha}} \{\boldsymbol{\alpha} \cdot \mathbf{r}_{t+1} - \zeta_1^*(\boldsymbol{\alpha})\}$
16: $\boldsymbol{\beta}_{t+1} = \arg\min_{\boldsymbol{\beta}} \{\boldsymbol{\beta} \cdot \mathbf{r}_{t+1} - \zeta_2^*(\boldsymbol{\beta})\}$
17: $\boldsymbol{\gamma}_{t+1} = \arg\min_{\boldsymbol{\gamma}} \{\boldsymbol{\gamma} \cdot \mathbf{q}_{t+1} - \Psi^*(\boldsymbol{\gamma})\}$
18: $t \leftarrow t + 1$
19: **end while**
20: **return** $\overline{\mathbf{w}} = \frac{1}{t} \sum_{\tau=1}^{t} \mathbf{w}_\tau$

We are now in a position to present our algorithm **NEMSIS** for stochastic optimization of nested concave functions. Algorithm 1 gives an outline of the technique. We note that a direct application of traditional stochastic optimization techniques [31] to such nested performance measures as those considered here is not possible as discussed before. **NEMSIS**, overcomes these challenges by exploiting the nested dual structure of the performance measure by carefully balancing updates at the inner and outer levels.

At every time step, **NEMSIS** performs four very cheap updates. The first update is a *primal* ascent update to the model vector which takes a weighted stochastic gradient descent step. Note that this step involves a *projection* step to the set of model vectors $\mathcal{W}$ denoted by $\Pi_{\mathcal{W}}(\cdot)$. In our experiments $\mathcal{W}$ was defined to be the set of all Euclidean norm-bounded vectors so that projection could be effected using Euclidean normalization which can be done in $\mathcal{O}(d)$ time if the model vectors are d-dimensional.

The weights of the descent step are decided by the dual parameters of the functions Ψ, ζ_1, and ζ_2. Then **NEMSIS** updates the dual variables in three simple steps. In fact line numbers 15-17 can be executed in closed form (see Table 1) for all the performance measures we see here which allows for very rapid updates. See Appendix A for the simple derivations.

Below we state the convergence proof for **NEMSIS**. We note that despite the complicated nature of the performance measures being tackled, **NEMSIS** is still able to recover the optimal rate of convergence known for stochastic optimization routines. We refer the reader to Appendix B for a proof of this theorem. The proof requires a careful analysis of the primal and dual update steps at different levels and tying the updates together by taking into account the nesting structure of the performance measure.

Theorem 3. *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 Algorithm 1 be executed with step sizes $\eta_t = \Theta(1/\sqrt{t})$ with a nested concave performance measure $\Psi(\zeta_1(\cdot), \zeta_2(\cdot))$. Then, for some universal constant C, 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 $1 - \delta$,*

$$\mathcal{P}_{(\Psi, \zeta_1, \zeta_2)}(\overline{\mathbf{w}}) \geq \sup_{\mathbf{w}^* \in \mathcal{W}} \mathcal{P}_{(\Psi, \zeta_1, \zeta_2)}(\mathbf{w}^*) - C_{\Psi, \zeta_1, \zeta_2, r} \cdot \left(\frac{\log \frac{1}{\delta}}{\sqrt{T}} \right),$$

Table 2: List of pseudo-concave performance measures and their canonical expressions in terms of the confusion matrix $\Psi(P, N)$. Note that p and n denote the proportion of positives and negatives in the population.

Name	Expression	$\mathcal{P}_{\text{quant}}(P, N)$	$\mathcal{P}_{\text{class}}(P, N)$
CQReward	$\frac{\text{BA}}{2-\text{NSS}}$	$1+(p(1-P)-n(1-N))^2$	$\frac{P+N}{2}$
BKReward	$\frac{\text{BA}}{1+\text{KLD}}$	KLD: see Table 1	$\frac{P+N}{2}$

where $C_{\Psi,\zeta_1,\zeta_2,r} = C(L_\Psi(L_r + B_r)(L_{\zeta_1} + L_{\zeta_2}))$ for a universal constant C and L_g denotes the Lipschitz constant of the function g.

Related work of Narasimhan et al : Narasimhan *et al* [25] proposed an algorithm **SPADE** which offered stochastic optimization of concave performance measures. We note that although the performance measures considered here are indeed concave, it is difficult to apply **SPADE** to them directly since **SPADE** requires computation of gradients of the Fenchel dual of the function $\mathcal{P}_{(\Psi,\zeta_1,\zeta_2)}$ which are difficult to compute given the nested structure of this function. **NEMSIS**, on the other hand, only requires the duals of the individual functions Ψ, ζ_1, and ζ_2 which are much more accessible. Moreover, **NEMSIS** uses a much simpler dual update which does not involve any parameters and, in fact, has a closed form solution in all our cases. **SPADE**, on the other hand, performs dual gradient descent which requires a fine tuning of yet another step length parameter. A third benefit of **NEMSIS** is that it achieves a logarithmic regret with respect to its dual updates (see the proof of Theorem 3) whereas **SPADE** incurs a polynomial regret due to its gradient descent-style dual update.

4.2 Pseudo-concave Performance Measures

The next class of performance measures we consider can be expressed as a ratio of a quantification and a classification performance measure. More formally, given a *convex* quantification performance measure $\mathcal{P}_{\text{quant}}$ and a *concave* classification performance measure $\mathcal{P}_{\text{class}}$, we can define a performance measure

$$\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w}) = \frac{\mathcal{P}_{\text{class}}(\mathbf{w})}{\mathcal{P}_{\text{quant}}(\mathbf{w})},$$

We assume that both the performance measures, $\mathcal{P}_{\text{quant}}$ and $\mathcal{P}_{\text{class}}$, are positive valued. Such performance measures can be very useful in allowing a system designer to balance classification and quantification performance. Moreover, the form of the measure allows an enormous amount of freedom in choosing the quantification and classification performance measures. Examples of such performance measures include the CQReward and the BKReward measures. These were introduced in Section 3.1 and are represented in their canonical forms in Table 2.

Performance measures, constructed the way described above, with a ratio of a concave over a convex measures, are called *pseudo-concave* measures. This is because, although these functions are not concave, their level sets are still convex which makes it possible to optimize them efficiently. To see the intuition behind this, we need to introduce the notion of the *valuation function* corresponding to the performance measure. As a passing note, we remark that because of the non-concavity of these performance measures, **NEMSIS** cannot be applied here.

Definition 4 (Valuation Function)**.** *The valuation of a pseudo-concave performance measure $\mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w}) = \frac{\mathcal{P}_{class}(\mathbf{w})}{\mathcal{P}_{quant}(\mathbf{w})}$ at any level $v > 0$, is defined as*

$$V(\mathbf{w}, v) = \mathcal{P}_{class}(\mathbf{w}) - v \cdot \mathcal{P}_{quant}(\mathbf{w})$$

It can be seen that the valuation function defines the level sets of the performance measure. To see this, notice that due to the positivity of the functions $\mathcal{P}_{\text{quant}}$ and $\mathcal{P}_{\text{class}}$, we can have $\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w}) \geq v$ iff $V(\mathbf{w}, v) \geq 0$. However, since $\mathcal{P}_{\text{class}}$ is concave, $\mathcal{P}_{\text{quant}}$ is convex, and $v > 0$, $V(\mathbf{w}, v)$ is a concave function of $\mathbf{w}$.

This close connection between the level sets and notions of valuation functions have been exploited before to give optimization algorithms for pseudo-linear performance measures such as the F-measure [25, 27]. These approaches treat the valuation function as some form of proxy or surrogate for the original performance measure and optimize it in hopes of making progress with respect to the original measure.

Algorithm 2 CAN: Concave AlternatioN

Require: Objective $\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}$, model space $\mathcal{W}$, tolerance ϵ
Ensure: An ϵ-optimal classifier $\mathbf{w} \in \mathcal{W}$
1: Construct the valuation function V
2: $\mathbf{w}_0 \leftarrow \mathbf{0}, t \leftarrow 1$
3: **while** $v_t > v_{t-1} + \epsilon$ **do**
4: $\quad \mathbf{w}_{t+1} \leftarrow \arg\max_{\mathbf{w}\in\mathcal{W}} V(\mathbf{w}, v_t)$
5: $\quad v_{t+1} \leftarrow \arg\max_{v>0} v$ such that $V(\mathbf{w}_{t+1}, v) \geq v$
6: $\quad t \leftarrow t + 1$
7: **end while**
8: **return** $\mathbf{w}_t$

Taking this approach with our performance measures yields a very natural algorithm for optimizing pseudo-concave measures which we outline in the **CAN** algorithm Algorithm 2. **CAN** repeatedly trains models to optimize their valuations at the current level, then upgrades the level itself. Notice that step 4 in the algorithm is a concave maximization problem over a convex set, something that can be done using a variety of methods – in the following we will see how **NEMSIS** can be used to implement this step. Also notice that step 5 can, by the definition of the valuation function, be carried out by simply setting $v_{t+1} = \mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w}_{t+1})$.

It turns out that **CAN** has a linear rate of convergence for well-behaved performance measures. The next result formalizes this statement. We note that this result is similar to the one arrived by [25] but only for *pseudo-linear* functions.

Theorem 5. *Suppose we execute Algorithm 2 with a pseudo-concave performance measure $\mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}$ such that the quantification performance measure always takes values in the range $[m, M]$, where $M > m > 0$. Let $\mathcal{P}^* := \sup_{\mathbf{w}\in\mathcal{W}} \mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w})$ be the optimal performance level and $\Delta_t = \mathcal{P}^* - \mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w}_t)$ be the excess error for the model $\mathbf{w}_t$ generated at time t. Then, for every $t > 0$, we have $\Delta_t \leq \Delta_0 \cdot \left(1 - \frac{m}{M}\right)^t$.*

We refer the reader to Appendix C for a proof of this theorem. This theorem generalizes the result of [25] to the more general case of pseudo-concave functions. Note that for the pseudo-concave functions defined in Table 2, care is taken to ensure that the quantification performance measure satisfies $m > 0$.

A drawback of **CAN** is that it cannot operate in streaming data settings and requires a concave optimization oracle. However, we notice that for the performance measures in Table 2, the valuation function is always at least a nested concave function. This motivates us to use **NEMSIS** to solve the inner optimization problems in an online fashion. Combining this with an online technique to approximately execute step 5 of of the **CAN** and gives us the **SCAN** algorithm, outlined in Algorithm 3.

Thoerem 6 shows that **SCAN** enjoys a convergence rate similar to that of **NEMSIS**. Indeed, **SCAN** is able to guarantee an ϵ-approximate solution after witnessing $\widetilde{\mathcal{O}}\left(1/\epsilon^2\right)$ samples which is equivalent to a convergence rate of $\widetilde{\mathcal{O}}\left(1/\sqrt{T}\right)$. The proof of this result is obtained by showing that **CAN** is robust to approximate solutions to the inner optimization problems. We refer the reader to Appendix D for a proof of this theorem.

Theorem 6. *Suppose we execute Algorithm 3 with a pseudo-concave performance measure $\mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}$ such that $\mathcal{P}_{quant}$ always takes values in the range $[m, M]$ with $m > 0$, with epoch lengths $s_e, s'_e = \frac{C_{\Psi,\zeta_1,\zeta_2,r}}{m^2}\left(\frac{M}{M-m}\right)^{2e}$ following a geometric rate of increase, where the constant $C_{\Psi,\zeta_1,\zeta_2,r}$ is the effective constant for the **NEMSIS** analysis (Theorem 3) for the inner invocations of **NEMSIS** in **SCAN**. Also let the excess error for the model $\mathbf{w}_e$ generated after e epochs be denoted by $\Delta_e = \mathcal{P}^* - \mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w}_e)$. Then after $e = \mathcal{O}\left(\log\left(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\right)\right)$ epochs, we can ensure with probability at least $1 - \delta$ that $\Delta_e \leq \epsilon$. Moreover, the number of samples consumed till this point, ignoring universal constants, is at most $\frac{C^2_{\Psi,\zeta_1,\zeta_2,r}}{\epsilon^2}\left(\log\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right)\log^4\frac{1}{\epsilon}$.*

Algorithm 3 SCAN: Stochastic Concave AlternatioN

Require: Objective $\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}$, model space $\mathcal{W}$, step sizes η_t, epoch lengths s_e, s'_e
Ensure: Classifier $\mathbf{w} \in \mathcal{W}$

$v_0 \leftarrow 0, t \leftarrow 0, e \leftarrow 0, \mathbf{w}_0 \leftarrow \mathbf{0}$
repeat
  // `Learning phase`
  $\widetilde{\mathbf{w}} \leftarrow \mathbf{w}_e$
  while $t < s_e$ **do**
    Receive sample $(\mathbf{x}, y)$
    // **`NEMSIS`** `update with` $V(\cdot, v_e)$ `at time` t
    $\mathbf{w}_{t+1} \leftarrow$ **NEMSIS** $(V(\cdot, v_e), \mathbf{w}_t, (\mathbf{x}, y), t)$
    $t \leftarrow t + 1$
  end while
  $t \leftarrow 0, e \leftarrow e + 1, \mathbf{w}_{e+1} \leftarrow \widetilde{\mathbf{w}}$
  // `Level estimation phase`
  $v_+ \leftarrow 0, v_- \leftarrow 0$
  while $t < s'_e$ **do**
    Receive sample $(\mathbf{x}, y)$
    $v_y \leftarrow v_y + r^y(\mathbf{w}_e; \mathbf{x}, y)$ // `Collect rewards`
    $t \leftarrow t + 1$
  end while
  $t \leftarrow 0, v_e \leftarrow \frac{\mathcal{P}_{\text{class}}(v_+, v_-)}{\mathcal{P}_{\text{quant}}(v_+, v_-)}$
until stream is exhausted
return $\mathbf{w}_e$

Table 3: Statistics of data sets used.

Data Set	Data Points	Features	Positives
KDDCup08	102,294	117	0.61%
PPI	240,249	85	1.19%
CoverType	581,012	54	1.63%
ChemInfo	2142	55	2.33%
Letter	20,000	16	3.92%
IJCNN-1	141,691	22	9.57%
Adult	48,842	123	23.93%
Cod-RNA	488,565	8	33.3%

Related work of Narasimhan et al : Narasimhan *et al* [25] also proposed two algorithms **AMP** and **STAMP** which seek to optimize *pseudo-linear* performance measures. However, neither those algorithms nor their analyses transfer directly to the pseudo-concave setting. This is because, by exploiting the pseudo-linearity of the performance measure, **AMP** and **STAMP** are able to convert their problem to a sequence of cost-weighted optimization problems which are very simple to solve. This convenience is absent here and as mentioned above, even after creation of the valuation function, **SCAN** still has to solve a possibly nested concave minimization problem which it does by invoking the **NEMSIS** procedure on this inner problem. The proof technique used in [25] for analyzing **AMP** also makes heavy use of pseudo-linearity. The convergence proof of **CAN**, on the other hand, is more general and yet guarantees a linear convergence rate.

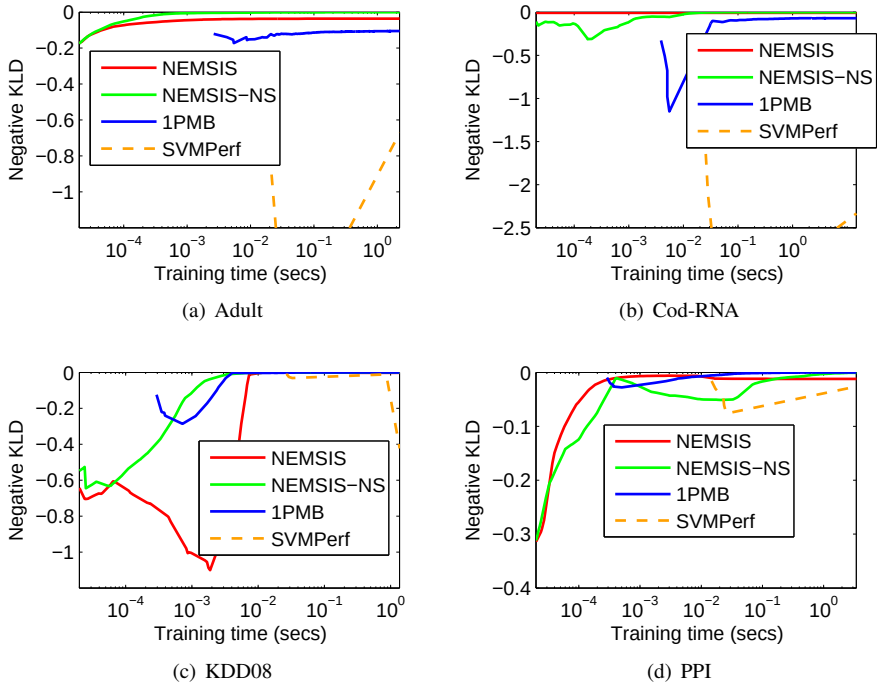
5 Experimental Results

We carried out an extensive set of experiments on diverse set of benchmark and real-world data to compare our proposed methods with other state-of-the-art approaches.

Data sets: We used the following benchmark data sets from the UCI machine learning repository : a) IJCNN, b) Covertype, c) Adult, d) Letters, and e) Cod-RNA. We also used the following three real-world data sets: a) Cheminformatics, a drug discovery data set from [19], b) 2008 KDD Cup challenge data set on breast cancer detection, and c) a data set pertaining to a protein-protein interaction (PPI) prediction task [28]. In each case, we used 70% of the data for training and the remaining for testing.

Methods: We compares our proposed **NEMSIS** and **SCAN** algorithms[3] against the state-of-the-art one-pass mini-batch stochastic gradient method (**1PMB**) of [20] and the SVM^{perf} technique of [18]. Both these techniques are capable of optimizing structural SVM surrogates of arbitrary performance measures and we modified their implementations to suitably adapt them to the performance measures considered here. The **NEMSIS** and **SCAN** implementations used the hinge-based concave surrogate.

Non-surrogate NS Approaches: We also experimented with a variant of the **NEMSIS** and **SCAN** algorithms, where the dual updates were computed using original count based TPR and TNR values, rather than surrogate reward functions. We refer to this version as **NEMSIS-NS**. We also developed a similar version of **SCAN** called **SCAN-NS** where the level estimation was performed using 0-1 TPR/TNR values. We empirically observed these non-surrogate versions of the algorithms to offer superior and more stable performance than the surrogate versions.

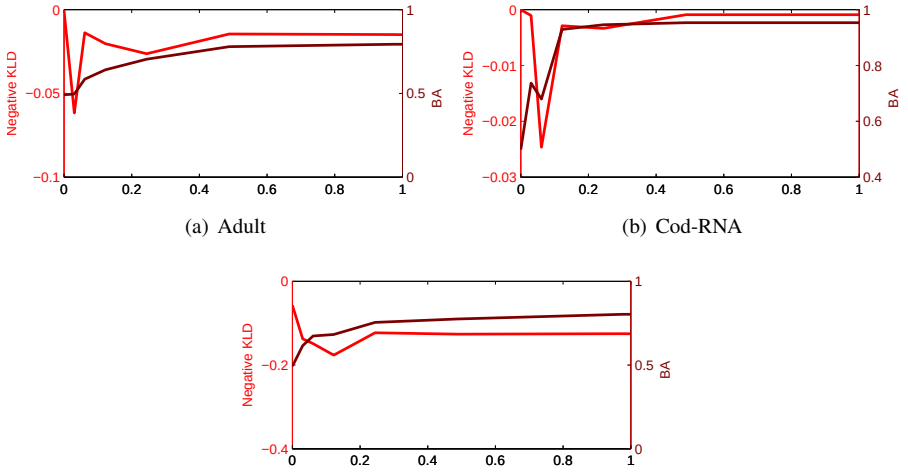


(a) Adult (b) Cod-RNA

(c) KDD08 (d) PPI

Figure 1: Experiments with **NEMSIS** on NegKLD: Plot of NegKLD as a function of training time.

[3]We will make code for our methods available publicly.



(a) Adult

(b) Cod-RNA

(c) Covtype

Figure 2: Experiments on **NEMSIS** with BAKLD: Plots of quantification and classification performance as CWeight is varied.

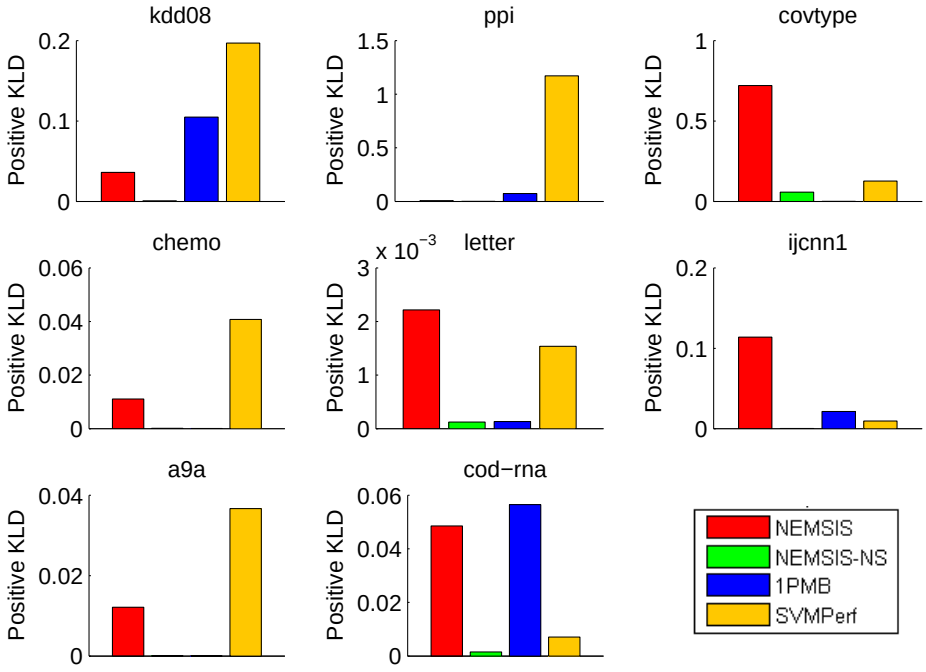
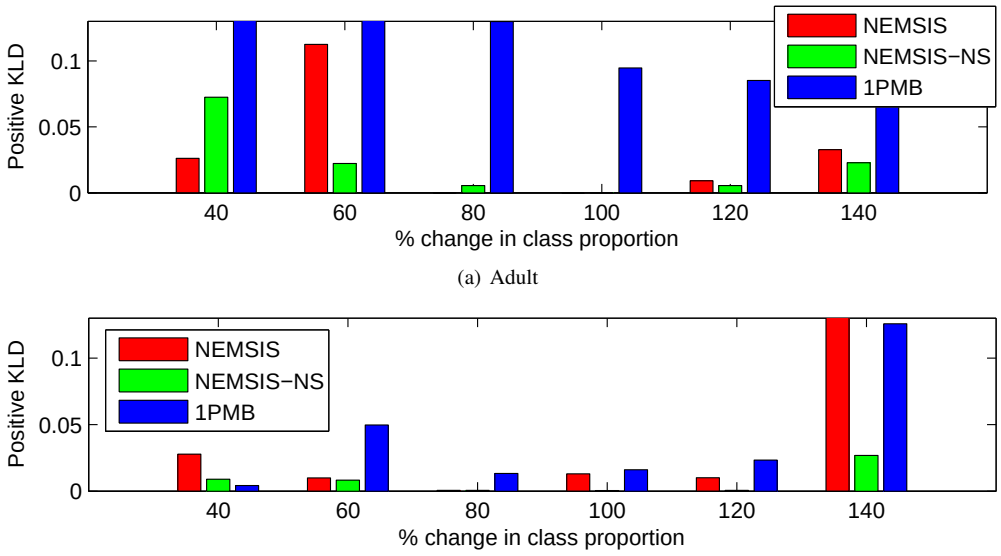


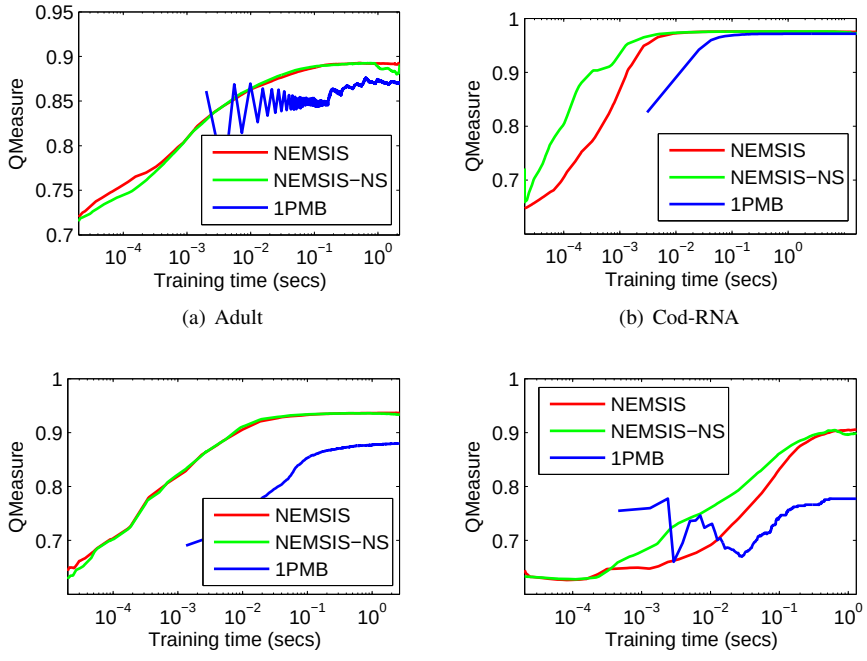
Figure 3: A comparison of the KLD performance of various methods on data sets with varying class proportions (see Table 4.2).



(a) Adult

(b) Letter

Figure 4: A comparison of the KLD performance of various methods when distribution drift is introduced in the test sets.



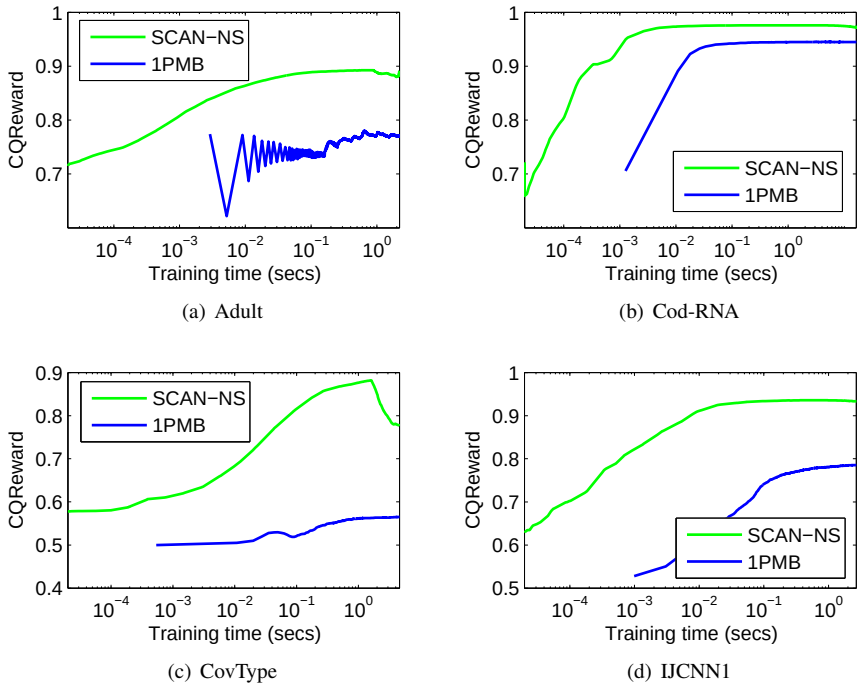
(a) Adult

(b) Cod-RNA

(c) IJCNN1

(d) KDD08

Figure 5: Experiments with **NEMSIS** on Q-measure: Plot of Q-measure performance as a function of time.



(a) Adult (b) Cod-RNA

(c) CovType (d) IJCNN1

Figure 6: Experiments with **SCAN** on CQreward: Plot of CQreward performance as a function of time.

Parameters: All parameters including step sizes, upper bounds on reward functions, regularization parameters, and projection radii were tuned from the values $\{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$ using a held-out portion of the training set treated as a validation set. For step sizes, the base step length η_0 was tuned from the above set and the step lengths were set to $\eta_t = \eta_0/\sqrt{t}$. In **1PMB**, we mimic the parameter setting in [20], setting the buffer size to 500 and the number of passes to 25.

Comparison on NegKLD: We first compare **NEMSIS-NS** and **NEMSIS** against the baselines **1PMB** and SVM^{perf} on several data sets on the negative KLD measure. The results are presented in Figure 1. It is clear that the proposed algorithms have comparable performance with significantly faster rate of convergence. Since SVM^{perf} is a batch/off-line method, it is important to clarify how it was compared against the other online methods. In this case, timers were embedded inside the SVM^{perf} code, and at regular intervals, the performance of the current model vector was evaluated. It is clear that SVM^{perf} is significantly slower and its behavior is quite erratic. The proposed methods are often faster than **1PMB**. On three of the four data sets **NEMSIS-NS** achieves a faster rate of convergence compared to **NEMSIS**.

Comparison on BAKLD: We also used the BAKLD performance measure to evaluate the trade-off **NEMSIS** offers between quantification and classification performance. The weighting parameter C in BAKLD (see Table 1), denoted here by CWeight to avoid confusion, was varied from 0 to 1 across a fine grid; for each value, **NEMSIS** was used to optimize BAKLD and its performance on BA and KLD were noted separately. In the results presented in Figure 2 for three data sets, notice that there is a sweet spot where the two tasks, i.e. quantification and classification simultaneously have good performance.

Comparison under varying class proportions: We next evaluated the robustness of the algorithms across data sets with varying different class proportions (see Table 4.2 for the dataset label proportions). In Figure 3, we plot positive KLD (smaller values are better) for the proposed and baseline methods for these diverse datasets. Again, it is clear that the **NEMSIS** family of algorithms of has better KLD performance compared to the baselines, demonstrating their versatility across a range of class distributions.

Comparison under varying drift: Next, we test the performance of the **NEMSIS** family of methods when there are drifts in class proportions between the train and test sets. In each case, we retain the original class proportion in the train set, and vary the class proportions in the test set, by suitably sampling from the original set of positive and negative test instances.[4] We have not included SVM^{perf} in these experiments as it took an inordinately long time to complete. As seen in Figure 4, on the Adult and Letter data set the **NEMSIS** family is fairly robust to small class drifts. As expected, when the class proportions change by a large amount in the test set (over 100 percent), all algorithms perform poorly.

Comparison on hybrid performance measures: Finally, we tested our methods in optimizing composite performance measures that strike a more nuanced trade-off between quantification and classification performance. Figures 5 contains results for the **NEMSIS** methods while optimizing Q-measure (see Table 1), and Figure 6 contains results for **SCAN-NS** while optimizing CQReward (see Table 2). The proposed methods are often significantly better than the baseline **1PMB** in terms of both accuracy and running time.

6 Conclusion

Quantification, the task of estimating class prevalence in problem settings subject to distribution drift, has emerged as an important problem in machine learning and data mining. Our discussion justified the necessity to design algorithms that exclusively solve the quantification task, with a special emphasis on performance measures such as the Kullback-Leibler divergence that is considered a de facto standard in the literature.

In this paper we proposed a family of algorithms **NEMSIS**, **CAN**, **SCAN**, and their non-surrogate versions, to address the online quantification problem. By abstracting NegKLD and other hybrid performance measures as nested concave or pseudo concave functions we designed provably correct and efficient algorithms for optimizing these performance measures in an online stochastic setting.

We validated our algorithms on several data sets under varying conditions, including class imbalance and distribution drift. The proposed algorithms demonstrate the ability to jointly optimize both quantification and classification tasks. To the best of our knowledge this is the first work which directly addresses the online quantification problem and as such, opens up novel application areas.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. PK thanks the Deep Singh and Daljeet Kaur Faculty Fellowship, and the Research-I Foundation at IIT Kanpur for support. SL acknowledges the support from 7Pixel S.r.l., SyChip Inc., and Murata Japan.

References

[1] Rocío Alaíz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing*, 74(16):2614–2623, 2011.

[4] More formally, we consider a setting where both the train and test sets are generated using the same conditional class distribution $\mathbf{P}(Y = 1 \,|X)$, but with different marginal distributions over instances $\mathbf{P}(X)$, and thus, have different class proportions. Further, in these experiments, we made a simplistic assumption that there is no label noise; hence for any instance $\mathbf{x}$, $\mathbf{P}(Y = 1 \,|\, X = \mathbf{x}) = 1$ or 0. Thus, we generated our test set with class proportion p' by simply setting $\mathbf{P}(X = \mathbf{x})$ to the following distribution: with probability p', sample a point uniformly from all points with $\mathbf{P}(Y = 1 \,|\, X = \mathbf{x}) = 1$, and with probability $1 - p'$, sample a point uniformly from all points with $\mathbf{P}(Y = 1 \,|\, X = \mathbf{x}) = 0$.

[2] Georgios Balikas, Ioannis Partalas, Eric Gaussier, Rohit Babbar, and Massih-Reza Amini. Efficient model selection for regularized classification by exploiting unlabeled data. In *Proceedings of the 14th International Symposium on Intelligent Data Analysis (IDA 2015)*, pages 25–36, Saint Etienne, FR, 2015.

[3] José Barranquero, Jorge Díez, and Juan José del Coz. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition*, 48(2):591–604, 2015.

[4] Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. Quantification via probability estimators. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010)*, pages 737–742, Sydney, AU, 2010.

[5] Nicoló Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. In *Proceedings of the 15th Annual Conference on Neural Information Processing Systems (NIPS 2001)*, pages 359–366, Vancouver, USA, 2001.

[6] Yee Seng Chan and Hwee Tou Ng. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006)*, pages 89–96, Sydney, AU, 2006.

[7] Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.

[8] Marthinus C. du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, UK, 2012.

[9] Andrea Esuli and Fabrizio Sebastiani. Sentiment quantification. *IEEE Intelligent Systems*, 25(4):72–75, 2010.

[10] Andrea Esuli and Fabrizio Sebastiani. Optimizing text quantifiers for multivariate loss functions. *ACM Transactions on Knowledge Discovery and Data*, 9(4):Article 27, 2015.

[11] George Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206, 2008.

[12] Wei Gao and Fabrizio Sebastiani. Tweet sentiment: From classification to quantification. In *Proceedings of the 7th International Conference on Advances in Social Network Analysis and Mining (ASONAM 2015)*, pages 97–104, Paris, FR, 2015.

[13] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, Bejing, CN, 2014.

[14] Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. Class distribution estimation based on the Hellinger distance. *Information Sciences*, 218:146–164, 2013.

[15] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic Regret Algorithms for Online Convex Optimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006)*, pages 499–513, Pittsburgh, USA, 2006.

[16] Daniel J. Hopkins and Gary King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229–247, 2010.

[17] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 377–384, Bonn, DE, 2005.

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

[19] Robert N. Jorissen and Michael K. Gilson. Virtual screening of molecular databases using a support vector machine. *Jounal of Chemical Information Modelling*, 45(3):549–561, 2005.

[20] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Online and stochastic gradient methods for non-decomposable loss functions. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS 2014)*, pages 694–702, Montreal, USA, 2014.

[21] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 189–198, Lille, FR, 2015.

[22] Gary King and Ying Lu. Verbal autopsy methods with multiple causes of death. *Statistical Science*, 23(1):78–91, 2008.

[23] David D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1995)*, pages 246–254, Seattle, USA, 1995.

[24] Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. Quantification trees. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM 2013)*, pages 528–536, Dallas, USA, 2013.

[25] Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 199–208, Lille, FR, 2015.

[26] Weike Pan, Erheng Zhong, and Qiang Yang. Transfer learning for text mining. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 223–258. Springer, Heidelberg, DE, 2012.

[27] Shameem P. Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing F-Measures by cost-sensitive classification. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS 2014)*, pages 2123–2131, Montreal, USA, 2014.

[28] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins*, 63:490–500, 2006.

[29] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, 2002.

[30] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic Convex Optimization. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, Montreal, CA, 2009.

[31] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. PEGASOS: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming, Series B*, 127(1):3–30, 2011.

[32] Jack Chongjie Xue and Gary M. Weiss. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2009)*, pages 897–906, Paris, FR, 2009.

[33] Constantin Zalinescu. *Convex Analysis in General Vector Spaces*. River Edge, NJ: World Scientific Publishing, 2002.

[34] Zhihao Zhang and Jie Zhou. Transfer estimation of evolving class priors in data stream classification. *Pattern Recognition*, 43(9):3151–3161, 2010.

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

A Deriving Updates for NEMSIS

The derivation of the closed form updates for steps 15-17 in the **NEMSIS** algorithm (see Algorithm 1) starts with the observation that in all the nested concave performance measures considered here, the outer and the inner concave functions, namely Ψ, ζ_1, ζ_2 are concave, continuous, and differentiable. The logarithm function is non-differentiable at 0 but the smoothing step (see Section refformulation) ensures that we will never approach 0 in our analyses or the execution of the algorithm. The derivations hinge on the following basic result from convex analysis [33]:

Lemma 7. *Let f be a closed, differentiable and concave function and f^* be its concave Fenchel dual. Then $\nabla f^* = (\nabla f)^{-1}$ i.e. for any $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{X}^*$ (the space of all linear functionals over $\mathcal{X}$), we have $\nabla f^*(\mathbf{u}) = \mathbf{x}$ iff $\nabla f(\mathbf{x}) = \mathbf{u}$.*

Using this result, we show how to derive the updates for $\boldsymbol{\gamma}$. The updates for $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ follow similarly. We have

$$\boldsymbol{\gamma}_t = \arg\min_{\boldsymbol{\gamma}} \{\boldsymbol{\gamma} \cdot \mathbf{q}_t - \Psi^*(\boldsymbol{\gamma})\}$$

By first order optimality conditions, we get that $\boldsymbol{\gamma}_t$ can minimize the function $h(\gamma) = \boldsymbol{\gamma} \cdot \mathbf{q}_t - \Psi^*(\boldsymbol{\gamma})$ only if $\mathbf{q}_t = \nabla\Psi^*(\boldsymbol{\gamma}_t)$. Using Lemma 7, we get $\boldsymbol{\gamma}_t = \nabla\Psi(\mathbf{q}_t)$. Using this technique, all the closed form expressions can be readily derived.

For the derivations of α, β for NegKLD, and the derivation of β for Q-measure, the derivations follow when we work with definitions of these performance measures with the TP and TN *counts* or cumulative surrogate reward values, rather than the TPR and TNR values and the average surrogate rewards.

B Proof of Theorem 3

We begin by observing the following general lemma regarding the follow the leader algorithm for strongly convex losses. This will be useful since steps 15-17 of Algorithm 1 are essentially executing follow the leader steps to decide the best value for the dual variables.

Lemma 8. *Suppose we have an action space $\mathcal{X}$ and execute the follow the leader algorithm on a sequence of loss functions $\ell_t : \mathcal{X} \to \mathbb{R}$, each of which is α-strongly convex and L-Lipschitz, then we have*

$$\sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \inf_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} \ell_t(\mathbf{x}) \leq \frac{L^2 \log T}{\alpha},$$

where $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{\tau=1}^{t} \ell_\tau(\mathbf{x})$ are the FTL plays.

Proof. By the standard forward regret analysis, we get

$$\sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \inf_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} \ell_t(\mathbf{x}) \leq \sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \sum_{t=1}^{T} \ell_t(\mathbf{x}_{t+1})$$

Now, by using the strong convexity of the loss functions, and the fact that the strong convexity property is additive, we get

$$\sum_{\tau=1}^{t-1} \ell_\tau(\mathbf{x}_{t+1}) \geq \sum_{\tau=1}^{t-1} \ell_\tau(\mathbf{x}_t) + \frac{\alpha(t-1)}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2$$
$$\sum_{\tau=1}^{t} \ell_\tau(\mathbf{x}_t) \geq \sum_{\tau=1}^{t} \ell_\tau(\mathbf{x}_{t+1}) + \frac{\alpha t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2,$$

which gives us

$$\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}) \geq \frac{\alpha}{2}(2t-1) \cdot \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2 .$$

However, we get $\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}) \leq L \cdot \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2$ by invoking Lipschitz-ness of the loss functions. This gives us

$$\|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2 \leq \frac{2L}{\alpha(2t-1)}.$$

This, upon applying Lipschitz-ness again, gives us

$$\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}) \leq \frac{2L^2}{\alpha(2t-1)}.$$

Summing over all the time steps gives us the desired result. □

For the rest of the proof, we shall use the shorthand notation that we used in Algorithm 1, i.e.

$$\begin{aligned}
\boldsymbol{\alpha}_t &= (\alpha_{t,1}, \alpha_{t,2}), && \text{(the dual variables for } \zeta_1) \\
\boldsymbol{\beta}_t &= (\beta_{t,1}, \beta_{t,2}), && \text{(the dual variables for } \zeta_2) \\
\boldsymbol{\gamma}_t &= (\gamma_{t,1}, \gamma_{t,2}), && \text{(the dual variables for } \Psi)
\end{aligned}$$

We will also use additional notation

$$\begin{aligned}
\mathbf{s}_t &= (r^+(\mathbf{w}_t; \mathbf{x}_t, y_t), r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)), \\
\mathbf{p}_t &= \left(\boldsymbol{\alpha}_t^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha}_t), \boldsymbol{\beta}_t^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta}_t)\right), \\
\boldsymbol{\ell}_t(\mathbf{w}) &= (r^+(\mathbf{w}; \mathbf{x}_t, y_t), r^-(\mathbf{w}; \mathbf{x}_t, y_t)) \\
R(\mathbf{w}) &= (P(\mathbf{w}), N(\mathbf{w}))
\end{aligned}$$

Note that $\boldsymbol{\ell}_t(\mathbf{w}_t) = \mathbf{s}_t$. We now define a quantity that we shall be analyzing to obtain the convergence bound

$$(A) = \sum_{t=1}^{T} (\boldsymbol{\gamma}_t^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma}_t))$$

Now, since Ψ is β_Ψ-smooth and concave, by Theorem 2, we know that Ψ^* is $\frac{1}{\beta}$-strongly concave. However that means that the loss function $g_t(\boldsymbol{\gamma}) := \boldsymbol{\gamma}^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma})$ is $\frac{1}{\beta}$-strongly convex. Now Algorithm 1 (step 17) implements

$$\boldsymbol{\gamma}_t = \arg\min_{\boldsymbol{\gamma}} \left\{ \boldsymbol{\gamma} \cdot \mathbf{q}_t - \Psi^*(\boldsymbol{\gamma}) \right\},$$

where $\mathbf{q}_t = \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{p}_\tau$ (see steps 7, 10, 13 that update $\mathbf{q}_t$). Notice that this is identical to the FTL algorithm with the losses $g_t(\boldsymbol{\gamma}) = \mathbf{p}_t \cdot \boldsymbol{\gamma} - \Psi^*(\boldsymbol{\gamma})$ which are strongly convex, and can be shown to be $(B_r(L_{\zeta_1} + L_{\zeta_2}))$-Lipschitz, neglecting universal constants. Thus, by an application of Lemma 8, we get, upto universal constants

$$(A) \leq \inf_{\boldsymbol{\gamma}} \left\{ \sum_{t=1}^{T} (\boldsymbol{\gamma}^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma})) \right\} + \beta_\Psi (B_r(L_{\zeta_1} + L_{\zeta_2}))^2 \log T$$

The same technique, along with the observation that steps 15 and 16 of Algorithm 1 also implement the FTL algorithm, can be used to get the following results upto universal constants

$$\sum_{t=1}^{T} (\boldsymbol{\alpha}_t^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha}_t)) \leq \inf_{\boldsymbol{\alpha}} \left\{ \sum_{t=1}^{T} (\boldsymbol{\alpha}^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha})) \right\} + \beta_{\zeta_1} (B_r L_{\zeta_1})^2 \log T,$$

and

$$\sum_{t=1}^{T}(\boldsymbol{\beta}_t^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta}_t)) \leq \inf_{\boldsymbol{\beta}} \left\{ \sum_{t=1}^{T}(\boldsymbol{\beta}^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta})) \right\} + \beta_{\zeta_2}(B_r L_{\zeta_2})^2 \log T.$$

This gives us, for

$$\Delta_1 = \beta_\Psi (B_r(L_{\zeta_1} + L_{\zeta_2}))^2 + \beta_{\zeta_1}(B_r L_{\zeta_1})^2 + \beta_{\zeta_2}(B_r L_{\zeta_2})^2,$$

$$\begin{aligned}
(A) &\leq \inf_{\boldsymbol{\gamma}} \left\{ \sum_{t=1}^{T}(\boldsymbol{\gamma}^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma})) \right\} + \Delta_1 \log T \\
&= \inf_{\boldsymbol{\gamma}} \left\{ \gamma_1 \sum_{t=1}^{T}(\boldsymbol{\alpha}_t^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_2 \sum_{t=1}^{T}(\boldsymbol{\beta}_t^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta}_t)) - \Psi^*(\boldsymbol{\gamma}) \right\} + \Delta_1 \log T \\
&\leq \inf_{\boldsymbol{\gamma},\boldsymbol{\alpha},\boldsymbol{\beta}} \left\{ \gamma_1 \sum_{t=1}^{T}(\boldsymbol{\alpha}^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha})) + \gamma_2 \sum_{t=1}^{T}(\boldsymbol{\beta}^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta})) - \Psi^*(\boldsymbol{\gamma}) \right\} + \Delta_1 \log T
\end{aligned}$$

Now, because of the stochastic nature of the samples, we have

$$\mathbb{E}\left[\!\left[\mathbf{s}_t | \left\{(\mathbf{x}_\tau, y_\tau)\right\}_{\tau=1}^{t-1} \right]\!\right] = \mathbb{E}\left[\!\left[\boldsymbol{\ell}_t(\mathbf{w}_t) | \left\{(\mathbf{x}_\tau, y_\tau)\right\}_{\tau=1}^{t-1} \right]\!\right] = R(\mathbf{w}_t)$$

$$\begin{aligned}
\frac{(A)}{T} &\leq \inf_{\boldsymbol{\gamma}} \left\{ \gamma_1 \inf_{\boldsymbol{\alpha}} \left\{ \boldsymbol{\alpha}^\top \left(\frac{1}{T}\sum_{t=1}^{T} R(\mathbf{w}_t) \right) - \zeta_1^*(\boldsymbol{\alpha}) \right\} \right. \\
&\qquad \left. + \left\{ \gamma_2 \inf_{\boldsymbol{\beta}} \left\{ \boldsymbol{\beta}^\top \left(\frac{1}{T}\sum_{t=1}^{T} R(\mathbf{w}_t) \right) - \zeta_2^*(\boldsymbol{\beta}) \right\} - \Psi^*(\boldsymbol{\gamma}) \right\} \right. \\
&\quad + \frac{\Delta_1}{T}\left(\log T + \log \frac{1}{\delta} \right) \\
&= \inf_{\boldsymbol{\gamma}} \left(\gamma_1 \zeta_1 \left(\frac{1}{T}\sum_{t=1}^{T} R(\mathbf{w}_t) \right) + \gamma_2 \zeta_2 \left(\frac{1}{T}\sum_{t=1}^{T} R(\mathbf{w}_t) \right) - \Psi^*(\boldsymbol{\gamma}) \right) \\
&\quad + \frac{\Delta_1}{T}\left(\log T + \log \frac{1}{\delta} \right) \\
&\leq \inf_{\boldsymbol{\gamma}} \left(\gamma_1 \zeta_1(R(\overline{\mathbf{w}})) + \gamma_2 \zeta_2(R(\overline{\mathbf{w}})) - \Psi^*(\boldsymbol{\gamma}) \right) + \frac{\Delta_1 \log \frac{T}{\delta}}{T} \\
&= \Psi(\zeta_1(R(\overline{\mathbf{w}})), \zeta_2(R(\overline{\mathbf{w}}))) + \frac{\Delta_1 \log \frac{T}{\delta}}{T},
\end{aligned}$$

where the second last step follows from the Jensen's inequality, the concavity of the functions $P(\mathbf{w})$ and $N(\mathbf{w})$, and the assumption that ζ_1 and ζ_2 are increasing functions of both their arguments. Thus, we have, with probability at least $1 - \delta$,

$$(A) \leq T \cdot \Psi(\zeta_1(R(\overline{\mathbf{w}})), \zeta_2(R(\overline{\mathbf{w}}))) + \Delta_1 \log \frac{T}{\delta}$$

Note that this is a much stronger bound than what Narasimhan *et al* [25] obtain for their gradient descent based dual updates. This, in some sense, establishes the superiority of the follow-the-leader type algorithms used by **NEMSIS**.

$$h_t(\mathbf{w}) = \gamma_{t,1}(\boldsymbol{\alpha}_t^\top \boldsymbol{\ell}_t(\mathbf{w}) - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_{t,2}(\boldsymbol{\beta}_t^\top \boldsymbol{\ell}_t(\mathbf{w}) - \zeta_2^*(\boldsymbol{\beta}_t)) - \Psi^*(\boldsymbol{\gamma}_t)$$

Since the functions $h_t(\cdot)$ are concave and $(L_\Psi L_r(L_{\zeta_1} + L_{\zeta_2}))$-Lipschitz (due to assumptions on the smoothness and values of the reward functions), the standard regret analysis for online gradient ascent (for example [35]) gives us the following bound on (A), ignoring universal constants

$$
\begin{aligned}
(A) &= \sum_{t=1}^{T} h_t(\mathbf{w}_t) \\
&= \sum_{t=1}^{T} \gamma_{t,1}(\boldsymbol{\alpha}_t^\top \boldsymbol{\ell}_t(\mathbf{w}_t) - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_{t,2}(\boldsymbol{\beta}_t^\top \boldsymbol{\ell}_t(\mathbf{w}_t) - \zeta_2^*(\boldsymbol{\beta}_t)) - \Psi^*(\boldsymbol{\gamma}_t) \\
&\geq \sum_{t=1}^{T} \gamma_{t,1}(\boldsymbol{\alpha}_t^\top \boldsymbol{\ell}_t(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_{t,2}(\boldsymbol{\beta}_t^\top \boldsymbol{\ell}_t(\mathbf{w}^*) - \zeta_2^*(\boldsymbol{\beta}_t)) \\
&\quad - \Psi^*(\boldsymbol{\gamma}_t) - \Delta_2\sqrt{T},
\end{aligned}
$$

where $\Delta_2 = (L_\Psi L_r(L_{\zeta_1} + L_{\zeta_2}))$. Note that the above results hold since we used step lengths $\eta_t = \Theta(1/\sqrt{t})$. To achieve the above bounds precisely, η_t will have to be tuned to the Lipschitz constant of the functions $h_t(\cdot)$ and for sake of simplicity we assume that the step lengths are indeed tuned so. We also assume, to get the above result , without loss of generality of course, that the model space $\mathcal{W}$ is the unit norm ball in $\mathbb{R}^d$. Applying a standard online-to-batch conversion bound (for example [5]), then gives us, with probability at least $1-\delta$,

$$
\begin{aligned}
\frac{(A)}{T} \geq\ & \underbrace{\frac{1}{T}\sum_{t=1}^{T} \gamma_{t,1}(\boldsymbol{\alpha}_t^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha}_t))}_{(B)} + \underbrace{\frac{1}{T}\sum_{t=1}^{T} \gamma_{t,2}(\boldsymbol{\beta}_t^\top R(\mathbf{w}^*) - \zeta_2^*(\boldsymbol{\beta}_t))}_{(C)} \\
& - \frac{1}{T}\sum_{t=1}^{T} \Psi^*(\boldsymbol{\gamma}_t) - \Delta_3 \frac{\log\frac{1}{\delta}}{\sqrt{T}},
\end{aligned}
$$

where $\Delta_3 = \Delta_2 + L_\Psi B_r(L_{\zeta_1} + L_{\zeta_2})$. Analyzing the expression (B) gives us

$$
\begin{aligned}
(B) &= \frac{1}{T}\sum_{t=1}^{T} \gamma_{t,1}(\boldsymbol{\alpha}_t^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha}_t)) \\
&= \frac{\sum_{t=1}^T \gamma_{t,1}}{T}\left(\left(\sum_{t=1}^{T} \frac{\gamma_{t,1}\boldsymbol{\alpha}_t}{\sum_{t=1}^T \gamma_{t,1}}\right)^\top R(\mathbf{w}^*) - \sum_{t=1}^{T} \frac{\gamma_{t,1}}{\sum_{t=1}^T \gamma_{t,1}} \zeta_1^*(\boldsymbol{\alpha}_t)\right) \\
&\geq \frac{\sum_{t=1}^T \gamma_{t,1}}{T}\left(\left(\sum_{t=1}^{T} \frac{\gamma_{t,1}\boldsymbol{\alpha}_t}{\sum_{t=1}^T \gamma_{t,1}}\right)^\top R(\mathbf{w}^*) - \zeta_1^*\left(\sum_{t=1}^{T} \frac{\gamma_{t,1}\boldsymbol{\alpha}_t}{\sum_{t=1}^T \gamma_{t,1}}\right)\right) \\
&\geq \frac{\sum_{t=1}^T \gamma_{t,1}}{T} \min_{\boldsymbol{\alpha}} \left\{\boldsymbol{\alpha}^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha})\right\} \\
&= \bar{\gamma}_1 \min_{\boldsymbol{\alpha}} \left\{\boldsymbol{\alpha}^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha})\right\} = \bar{\gamma}_1 \zeta_1(R(\mathbf{w}^*))
\end{aligned}
$$

A similar analysis for (C) follows and we get, ignoring universal constants,

$$
\begin{aligned}
\frac{(A)}{T} &\geq \bar{\gamma}_1\zeta_1(R(\mathbf{w}^*)) + \bar{\gamma}_2\zeta_2(R(\mathbf{w}^*)) - \frac{1}{T}\sum_{t=1}^{T} \Psi^*(\boldsymbol{\gamma}_t) - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}} \\
&\geq \bar{\gamma}_1\zeta_1(R(\mathbf{w}^*)) + \bar{\gamma}_2\zeta_2(R(\mathbf{w}^*)) - \Psi^*(\bar{\boldsymbol{\gamma}}) - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}} \\
&\geq \min_{\boldsymbol{\gamma}} \{\gamma_1\zeta_1(R(\mathbf{w}^*)) + \gamma_1\zeta_2(R(\mathbf{w}^*)) - \Psi^*(\boldsymbol{\gamma})\} - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}}
\end{aligned}
$$

$$= \Psi(\zeta_1(R(\mathbf{w}^*)), \zeta_2(R(\mathbf{w}^*))) - \Delta_3 \frac{\log \frac{1}{\delta}}{\sqrt{T}}$$

Thus, we have with probability at least $1 - \delta$,

$$(A) \geq T \cdot \Psi(\zeta_1(R(\mathbf{w}^*)), \zeta_2(R(\mathbf{w}^*))) - \Delta_3 \log \frac{1}{\delta} \sqrt{T}$$

Combining the upper and lower bounds on (A) finishes the proof since $\Delta_3 \log \frac{1}{\delta} \sqrt{T}$ overwhelms the term $\Delta_1 \log \frac{T}{\delta}$.

C Proof of Theorem 5

We will prove the result by proving a sequence of claims. The first claim ensures that the distance to the optimum performance value is bounded by the performance value we obtain in terms of the valuation function at any step. For notational simplicity, we will use the shorthand $\mathcal{P}(\mathbf{w}) := \mathcal{P}_{(\mathcal{P}_{\text{quant}}, \mathcal{P}_{\text{class}})}(\mathbf{w})$.

Claim 9. *$\mathcal{P}^* := \sup_{\mathbf{w} \in \mathcal{W}} \mathcal{P}(\mathbf{w})$ be the optimal performance level. Also, define $e_t = V(\mathbf{w}_{t+1}, v_t)$. Then, for any t, we have*

$$\mathcal{P}^* - \mathcal{P}(\mathbf{w}_t) \leq \frac{e_t}{m}$$

Proof. We will prove the result by contradiction. Suppose $\mathcal{P}^* > \mathcal{P}(\mathbf{w}_t) + \frac{e_t}{m}$. Then there must exist some $\tilde{\mathbf{w}} \in \mathcal{W}$ such that

$$\mathcal{P}(\tilde{\mathbf{w}}) = \frac{e_t}{m} + \mathcal{P}(\mathbf{w}_t) + e' = \frac{e_t}{m} + v_t + e' =: v',$$

where $e' > 0$. Note that the above uses the fact that we set $v_t = \mathcal{P}(\mathbf{w}_t)$. Then we have

$$V(\tilde{\mathbf{w}}, v_t) - e_t = \mathcal{P}_{\text{class}}(\tilde{\mathbf{w}}) - v_t \cdot \mathcal{P}_{\text{quant}}(\tilde{\mathbf{w}}) - e_t.$$

Now since $\mathcal{P}(\tilde{\mathbf{w}}) = v'$, we have $\mathcal{P}_{\text{class}}(\tilde{\mathbf{w}}) - v' \cdot \mathcal{P}_{\text{quant}}(\tilde{\mathbf{w}}) = 0$ which gives us

$$V(\tilde{\mathbf{w}}, v_t) - e_t = \left(\frac{e_t}{m} + e'\right) \mathcal{P}_{\text{quant}}(\tilde{\mathbf{w}}) - e_t \geq \left(\frac{e_t}{m} + e'\right) m - e_t > 0.$$

But this contradicts the fact that $\max_{\mathbf{w} \in \mathcal{W}} V(\mathbf{w}, v_t) = e_t$ which is ensured by step 4 of Algorithm 2. This completes the proof. □

The second claim then establishes that in case we do get a large performance value in terms of the valuation function at any time step, the next iterate will have a large leap in performance in terms of the original performance function $\mathcal{P}$.

Claim 10. *For any time instant t we have*

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq \mathcal{P}(\mathbf{w}_t) + \frac{e_t}{M}$$

Proof. By our definition, we have $V(\mathbf{w}_{t+1}, v_t) = e_t$. This gives us

$$\begin{aligned} \frac{\mathcal{P}_{\text{class}}(\mathbf{w}_{t+1})}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})} - \left(v_t + \frac{e_t}{M}\right) &\geq \frac{\mathcal{P}_{\text{class}}(\mathbf{w}_{t+1})}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})} - \left(v_t + \frac{e_t}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})}\right) \\ &= \frac{v_t \cdot \mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})} - v_t = 0, \end{aligned}$$

which proves the result. □

We are now ready to establish the convergence proof. Let $\Delta_t = \mathcal{P}^* - \mathcal{P}(\mathbf{w}_t)$. Then we have, by Claim 9

$$e_t \geq m \cdot \Delta_t,$$

and also

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq \mathcal{P}(\mathbf{w}_t) + \frac{e_t}{M},$$

by Claim 10. Subtracting both sides of the above equation from $\mathcal{P}^*$ gives us

$$\begin{aligned}\Delta_{t+1} &= \Delta_t - \frac{e_t}{M} \\ &\leq \Delta_t - \frac{m}{M} \cdot \Delta_t = \left(1 - \frac{m}{M}\right) \cdot \Delta_t,\end{aligned}$$

which concludes the convergence proof.

D Proof of Theorem 6

To prove this theorem, we will first show that the **CAN** algorithm is robust to imprecise updates. More precisely, we will assume that Algorithm 2 only ensures that in step 4 we have

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

where $\epsilon_t > 0$ and step 5 only ensures that

$$v_t = \mathcal{P}(\mathbf{w}_t) + \delta_t,$$

where δ_t may be positive or negative. For this section, we will redefine

$$e_t = \max_{\mathbf{w} \in \mathcal{W}} V(\mathbf{w}, v_t)$$

since we can no longer assume that $V(\mathbf{w}_{t+1}, v_t) = e_t$. Note that if v_t is an unrealizable value, i.e. for no predictor $\mathbf{w} \in \mathcal{W}$ is $\mathcal{P}(\mathbf{w}) \geq v_t$, then we have $e_t < 0$. Having this we establish the following results:

Lemma 11. *Given the previous assumptions on the imprecise execution of Algorithm 2, the following is true*

1. *If* $\delta_t \leq 0$ *then* $e_t \geq 0$
2. *If* $\delta_t > 0$ *then* $e_t \geq -\delta_t \cdot M$
3. *We have* $\mathcal{P}^* < v_t$ *iff* $e_t < 0$
4. *If* $e_t \geq 0$ *then* $e_t \geq m(\mathcal{P}^* - v_t)$
5. *If* $e_t < 0$ *then* $e_t \geq M(\mathcal{P}^* - v_t)$
6. *If* $V(\mathbf{w}, v) = e$ *for* $e \geq 0$ *then* $\mathcal{P}(\mathbf{w}) \geq v + \frac{e}{M}$
7. *If* $V(\mathbf{w}, v) = e$ *for* $e < 0$ *then* $\mathcal{P}(\mathbf{w}) \geq v + \frac{e}{m}$

Proof. We prove the parts separately below

1. Since $\delta_t < 0$, there exists some $\mathbf{w} \in \mathcal{W}$ such that $\mathcal{P}(\mathbf{w}) > v_t$. The result then follows.
2. If $v_t = \mathcal{P}(\mathbf{w}_t) + \delta_t$ then $V(\mathbf{w}_t, v_t) \geq -\delta_t \cdot M$ .The result then follows.
3. Had $e_t \geq 0$ been the case, we would have had, for some $\mathbf{w} \in \mathcal{W}$, $V(\mathbf{w}, v_t) \geq 0$ which would have implied $\mathcal{P}(\mathbf{w}) \geq v_t$ which contradicts $\mathcal{P}^* < v_t$. For the other direction, suppose $\mathcal{P}^* = \mathcal{P}(\mathbf{w}^*) = v_t + e'$ with $e' > 0$. Then we have $e_t = V(\mathbf{w}^*, v_t) > 0$ which contradicts $e_t < 0$.

4. Observe that the proof of Claim 9 suffices, by simply replacing $\mathcal{P}(\mathbf{w}_t)$ with v_t in the statement.

5. Assume the contrapositive that for some $\mathbf{w} \in \mathcal{W}$, we have $\mathcal{P}(\mathbf{w}) = v_t + \frac{e_t}{M} + e'$ where $e' > 0$. We can then show that $V(\mathbf{w}, v_t) = \left(\frac{e_t}{M} + e'\right) \mathcal{P}_{\text{quant}}(\mathbf{w}) \geq e_t + e' \cdot \mathcal{P}_{\text{quant}}(\mathbf{w}) > e_t$ which contradicts the definition of e_t. Note that since $e_t < 0$, we have $\frac{e_t}{M} \geq \frac{e_t}{\mathcal{P}_{\text{quant}}(\mathbf{w})}$ and we have $\mathcal{P}_{\text{quant}}(\mathbf{w}) \geq m > 0$.

6. Observe that the proof of Claim 10 suffices, by simply replacing $\mathcal{P}(\mathbf{w}_t)$ with v_t in the statement.

7. We have $\mathcal{P}_{\text{class}}(\mathbf{w}) - v \cdot \mathcal{P}_{\text{quant}}(\mathbf{w}) = e$. Dividing throughout by $\mathcal{P}_{\text{quant}}(\mathbf{w}) > 0$ and using $\frac{e}{\mathcal{P}_{\text{quant}}(\mathbf{w})} \geq \frac{e}{m}$ since $e < 0$ gives us the result.

This finishes the proofs. □

Using these results, we can now make the following claim on the progress made by **CAN** with imprecise updates.

Lemma 12. *Even if **CAN** is executed with noisy updates, at any time step t, we have*

$$\Delta_{t+1} \leq \left(1 - \frac{m}{M}\right) \Delta_t + \frac{M}{m} \cdot |\delta_t| + \frac{\epsilon_t}{m}.$$

Proof. We analyze time steps when $\delta_t \leq 0$ separately from time steps when $\delta_t < 0$.

Case 1: $\boldsymbol{\delta_t \leq 0}$ In these time steps, the method underestimates the performance of the current predictor but gives a legal i.e. realizable value of v_t. We first deduce that for these time steps, using Lemma 11 part 1, we have $e_t \geq 0$ and then using part 4, we have $e_t \geq m(\mathcal{P}^* - v_t)$. This combined with the identity $\mathcal{P}^* - v_t = \Delta_t - \delta_t$, gives us

$$e_t \geq m(\Delta_t - \delta_t)$$

Now we have, by definition, $V(\mathbf{w}_{t+1}, v_t) = e_t - \epsilon_t$ (note that both $e_t, \epsilon \geq 0$ in this case). The next steps depend on whether this quantity is positive or negative. If $\epsilon_t \leq e_t$, we apply Lemma 11 part 6 to get

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq v_t + \frac{e_t - \epsilon_t}{M},$$

which gives us upon using $\mathcal{P}^* - v_t = \Delta_t - \delta_t$ and $e_t \geq m(\Delta_t - \delta_t)$,

$$\Delta_{t+1} \leq \left(1 - \frac{m}{M}\right) \Delta_t - \left(1 - \frac{m}{M}\right) \delta_t + \frac{\epsilon_t}{M}$$

Otherwise if $\epsilon_t > e_t$ then we have actually made negative progress at this time step since $V(\mathbf{w}_{t+1}, v_t) < 0$. To safeguard us against how much we go back in terms of progress, we us Lemma 11 part 7 to guarantee

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq v_t + \frac{e_t - \epsilon_t}{m},$$

which gives us upon using $\mathcal{P}^* - v_t = \Delta_t - \delta_t$ and $e_t \geq m(\Delta_t - \delta_t)$,

$$\Delta_{t+1} \leq \frac{\epsilon_t}{m},$$

Note however, that we are bound by $\epsilon_t > e_t$ in the above statement. We now move on to analyze the second case.

Case 2: $\boldsymbol{\delta_t > 0}$ In these time steps, the method is overestimating the performance of the current predictor and runs a risk of giving a value of v_t that is unrealizable. We cannot hope to make much progress in these time steps. The following analysis simply safeguards us against too much deterioration. There are two subcases we explore here: first we look at the case where $v_t \leq \mathcal{P}^*$ i.e. v_t is still a legal, realizable performance value. In this case we continue to have $e_t \geq 0$ and the analysis of the previous case (i.e. $\delta_t \leq 0$) continues to apply.

However, if $v_t > \mathcal{P}^*$, we are setting an unrealizable value of v_t. Using Lemma 11 part 3 gives us $e_t < 0$ which, upon using part 5 of the lemma gives us

$$e_t \geq M(\mathcal{P}^* - v_t).$$

In this case, we have $V(\mathbf{w}_{t+1}, v_t) = e_t - \epsilon_t < 0$ since $e_t < 0$ and $\epsilon_t > 0$. Thus, using Lemma 11 part 7 gives us

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq v_t + \frac{e_t - \epsilon_t}{m}$$

which upon manipulation, as before, gives us

$$\Delta_{t+1} \leq \left(1 - \frac{M}{m}\right) \Delta_t + \left(\frac{M}{m} - 1\right) \delta_t + \frac{\epsilon_t}{m} \leq \left(\frac{M}{m} - 1\right) \delta_t + \frac{\epsilon_t}{m},$$

where the last step uses the fact that $\Delta_t \geq 0$ and $M \geq m$. Putting all these cases together and using the fact that the quantities $\Delta_t, \epsilon_t, |\delta_t|$ are always positive gives us

$$\begin{aligned}\Delta_{t+1} &\leq \left(1 - \frac{m}{M}\right) \Delta_t + \left(\frac{M^2 - m^2}{mM}\right) |\delta_t| + \frac{\epsilon_t}{m} \\ &\leq \left(1 - \frac{m}{M}\right) \Delta_t + \frac{M}{m} \cdot |\delta_t| + \frac{\epsilon_t}{m},\end{aligned}$$

which finishes the proof. □

From hereon simple manipulations similar to those used to analyze the **STAMP** algorithm in [25] can be used, along with the guarantees provided by Theorem 3 for the **NEMSIS** analysis to finish the proof of the result. We basically have to use the fact that the **NEMSIS** invocations in **SCAN** (Algorithm 3 line 8), as well as the performance estimation steps (Algorithm 3 lines 14-19) can be seen as executing noisy updates for the original **CAN** algorithm.