An Exact Algorithm for F-Measure Maximization

Krzysztof Dembczyński
Institute of Computing Science
Poznań University of Technology
Poznań, 60-695 Poland
kdembczynski@cs.put.poznan.pl

Willem Waegeman
Mathematical Modelling, Statistics
and Bioinformatics, Ghent University
Ghent, 9000 Belgium
willem.waegeman@ugent.be

Weiwei Cheng
Mathematics and Computer Science
Philipps-Universität Marburg
Marburg, 35032 Germany
cheng@mathematik.uni-marburg.de

Eyke Hüllermeier
Mathematics and Computer Science
Philipps-Universität Marburg
Marburg, 35032 Germany
eyke@mathematik.uni-marburg.de

Abstract

The F-measure, originally introduced in information retrieval, is nowadays routinely used as a performance metric for problems such as binary classification, multi-label classification, and structured output prediction. Optimizing this measure remains a statistically and computationally challenging problem, since no closed-form maximizer exists. Current algorithms are approximate and typically rely on additional assumptions regarding the statistical distribution of the binary response variables. In this paper, we present an algorithm which is not only computationally efficient but also exact, regardless of the underlying distribution. The algorithm requires only a quadratic number of parameters of the joint distribution (with respect to the number of binary responses). We illustrate its practical performance by means of experimental results for multi-label classification.

1 Introduction

While being rooted in information retrieval [1], the so-called F-measure is nowadays routinely used as a performance metric for different types of prediction problems, including binary classification, multi-label classification (MLC), and certain applications of structured output prediction, like text chunking and named entity recognition. Compared to measures like error rate in binary classification and Hamming loss in MLC, it enforces a better balance between performance on the minority and the majority class, respectively, and, therefore, it is more suitable in the case of imbalanced data. Given a prediction $\boldsymbol{h} = (h_1, \ldots, h_m) \in \{0,1\}^m$ of an m-dimensional binary label vector $\boldsymbol{y} = (y_1, \ldots, y_m)$ (e.g., the class labels of a test set of size m in binary classification or the label vector associated with a single instance in MLC), the F-measure is defined as follows:

$$F(\boldsymbol{y}, \boldsymbol{h}) = \frac{2 \sum_{i=1}^{m} y_i h_i}{\sum_{i=1}^{m} y_i + \sum_{i=1}^{m} h_i} \in [0,1] \ , \tag{1}$$

where $0/0 = 1$ by definition. This measure essentially corresponds to the harmonic mean of precision $prec$ and recall rec:

$$prec(\boldsymbol{y}, \boldsymbol{h}) = \frac{\sum_{i=1}^{m} y_i h_i}{\sum_{i=1}^{m} h_i}, \quad rec(\boldsymbol{y}, \boldsymbol{h}) = \frac{\sum_{i=1}^{m} y_i h_i}{\sum_{i=1}^{m} y_i} \ .$$

One can generalize the F-measure to a weighted harmonic average of these two values, but for the sake of simplicity, we stick to the unweighted mean, which is often referred to as the F1-score or the F1-measure.

Despite its popularity in experimental settings, only a few methods for training classifiers that directly optimize the F-measure have been proposed so far. In binary classification, the existing algorithms are extensions of support vector machines [2, 3] or logistic regression [4]. However, the most popular methods, including [5], rely on explicit threshold adjustment. Some algorithms have also been proposed for structured output prediction [6, 7, 8] and MLC [9, 10, 11]. In these two application domains, three different aggregation schemes of the F-measure can be distinguished, namely the instance-wise, the micro-, and the macro-averaging. One should carefully distinguish these versions, as algorithms optimized with a given objective are usually performing suboptimally for other (target) evaluation measures.

All the above algorithms intend to optimize the F-measure during the training phase. Conversely, in this article we rather investigate an orthogonal problem of inference from a probabilistic model. Modeling the ground-truth as a random variable $\boldsymbol{Y}$, i.e., assuming an underlying probability distribution $p(\boldsymbol{Y})$ on $\{0,1\}^m$, the prediction $\boldsymbol{h}_F^*$ that maximizes the expected F-measure is given by

$$\boldsymbol{h}_F^* = \operatorname*{arg\,max}_{\boldsymbol{h}\in\{0,1\}^m} \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{Y})}\left[F(\boldsymbol{y},\boldsymbol{h})\right] = \operatorname*{arg\,max}_{\boldsymbol{h}\in\{0,1\}^m} \sum_{\boldsymbol{y}\in\{0,1\}^m} p(\boldsymbol{Y}{=}\boldsymbol{y})\, F(\boldsymbol{y},\boldsymbol{h}). \tag{2}$$

As discussed in Section 2, this setting was mainly examined before by [12], under the assumption of independence of the Y_i, i.e., $p(\boldsymbol{Y}{=}\boldsymbol{y}) = \prod_{i=1}^m p_i^{y_i}(1-p_i)^{1-y_i}$ with $p_i{=}p(Y_i{=}1)$. Indeed, finding the maximizer (2) is in general a difficult problem. Apparently, there is no closed-form expression, and a brute-force search is infeasible (it would require checking all 2^m combinations of prediction vector $\boldsymbol{h}$). At first sight, it also seems that information about the entire joint distribution $p(\boldsymbol{Y})$ is needed to maximize the F-measure. Yet, as will be shown in this paper, the problem can be solved more efficiently. In Section 3, we present a general algorithm for maximizing the F-measure that requires only m^2+1 parameters of the joint distribution. If these parameters are given, the exact solution can be obtained in time $o(m^3)$. This result holds regardless of the underlying distribution. In particular, unlike algorithms such as [12], we do not require independence of the binary response variables (labels). While being natural for problems like binary classification, this assumption is indeed not tenable in domains like MLC and structured output prediction. A discussion of existing methods for F-measure maximization, along with results indicating their shortcomings, is provided in Section 2. An experimental comparison in the context of MLC is presented in Section 4.

2 Existing Algorithms for F-Measure Maximization

Current algorithms for solving (2) make different assumptions to simplify the problem. First of all, the algorithms operate on a constrained hypothesis space, sometimes justified by theoretical arguments. Secondly, they guarantee optimality only for specific distributions $p(\boldsymbol{Y})$.

2.1 Algorithms Based on Label Independence

By assuming independence of the random variables $Y_1, ..., Y_m$, the optimization problem (2) can be substantially simplified. It has been shown independently in [13] and [12] that the optimal solution always contains the labels with the highest marginal probabilities p_i, or no labels at all. As a consequence, only a few hypotheses $\boldsymbol{h}$ ($m{+}1$ instead of 2^m) need to be examined, and the computation of the expected F-measure can be performed in an efficient way.

Lewis [13] showed that the expected F-measure can be approximated by the following expression under the assumption of independence:[1]

$$\mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{Y})}\left[F(\boldsymbol{y},\boldsymbol{h})\right] \simeq \begin{cases} \prod_{i=1}^m (1-p_i), & \text{if } \boldsymbol{h}=\boldsymbol{0} \\ \frac{2\sum_{i=1}^m p_i h_i}{\sum_{i=1}^m p_i + \sum_{i=1}^m h_i}, & \text{if } \boldsymbol{h}\neq\boldsymbol{0} \end{cases}$$

This approximation is exact for $\boldsymbol{h}=\boldsymbol{0}$, while for $\boldsymbol{h}\neq\boldsymbol{0}$, an upper bound of the error can easily be determined [13].

Jansche [12], however, has proposed an exact procedure, called maximum expected utility framework (MEUF), that takes marginal probabilities $p_1, p_2, \ldots, p_m$ as inputs and solves (2) in time

[1] In the following, we denote $\boldsymbol{0}$ and $\boldsymbol{1}$ as vectors containing all zeros and ones, respectively.

$\mathcal{O}(m^4)$. He noticed that (2) can be solved via outer and inner maximization. Namely, (2) can be transformed into an inner maximization

$$\boldsymbol{h}^{(k)^*} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})} \left[F(\boldsymbol{y}, \boldsymbol{h}) \right], \tag{3}$$

where $H_k = \{\boldsymbol{h} \in \{0,1\}^m \mid \sum_{i=1}^m h_i = k\}$, followed by an outer maximization

$$\boldsymbol{h}_F^* = \operatorname*{arg\,max}_{\boldsymbol{h} \in \{\boldsymbol{h}^{(0)^*}, \ldots, \boldsymbol{h}^{(m)^*}\}} \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})} \left[F(\boldsymbol{y}, \boldsymbol{h}) \right]. \tag{4}$$

The outer maximization (4) can be done by simply checking all $m+1$ possibilities. The main effort is then devoted for solving the inner maximization (3). According to Theorem 2.1, to solve (3) for a given k, we need to check only one vector $\boldsymbol{h}$ in which $h_i = 1$ for the k labels with highest marginal probabilities p_i. The remaining problem is the computation of the expected F-measure in (3). This expectation cannot be computed naively, as the sum is over exponentially many terms. But the F-measure is a function of integer counts that are bounded, so it can normally only assume a much smaller number of distinct values. The cardinality of its domain is indeed exponential in m, but the cardinality of its range is polynomial in m, so the expectation can be computed in polynomial time. As a result, Jansche [12] obtains a procedure that is cubic in m for computing (3). He also presents approximate variants of this procedure, reducing its complexity from cubic to quadratic or even to linear. The results of the quadratic-time approximation, according to [12], are almost indistinguishable in practice from the exact algorithm; but still the overall complexity of the approach is $\mathcal{O}(m^3)$.

If the independence assumption is violated, the above methods may produce predictions being far away from the optimal one. The following result shows this concretely for the method of Jansche.[2]

Proposition 2.1. *Let $\boldsymbol{h}_J$ be a vector of predictions obtained by MEUF, then the worst-case regret converges to one in the limit of m, i.e.,*

$$\lim_{m \to \infty} \sup_{p} \left(\mathbb{E}_{\mathbf{Y}} \left[F(\mathbf{Y}, \boldsymbol{h}_F^*) - F(\mathbf{Y}, \boldsymbol{h}_J) \right] \right) = 1,$$

where the supremum is taken over all possible distributions $p(\boldsymbol{Y})$.

Additionally, one can easily construct families of probability distributions that obtain a relatively fast convergence rate as a function of m.

2.2 Algorithms Based on the Multinomial Distribution

Solving (2) becomes straightforward in the case of a specific distribution in which the probability mass is distributed over vectors $\boldsymbol{y}$ containing only a single positive label, i.e., $\sum_{i=1}^m y_i = 1$, corresponding to the multinomial distribution. This was studied in [14] in the setting of so-called *non-deterministic classification*.

Theorem 2.2 (Del Coz et al. [14])**.** *Denote by $\boldsymbol{y}(i)$ a vector for which $y_i = 1$ and all the other entries are zeros. Assume that $p(\boldsymbol{Y})$ is a joint distribution such that $p(\boldsymbol{Y} = \boldsymbol{y}(i)) = p_i$. The maximizer $\boldsymbol{h}_F^*$ of (2) consists of the k labels with the highest marginal probabilities, where k is the first integer for which*

$$\sum_{j=1}^{k} p_j \geq (1+k) p_{k+1};$$

if there is no such integer, then $\boldsymbol{h} = \mathbf{1}$.

2.3 Algorithms Based on Thresholding on Ordered Marginal Probabilities

Since all the methods so far rely on the fact that the optimal solution contains ones for the labels with the highest marginal probabilities (or consists of a vector of zeros), one may expect that thresholding on the marginal probabilities ($h_i = 1$ for $p_i \geq \theta$, and $h_i = 0$ otherwise) will provide a solution to

[2] Some of the proofs have been attached to the paper as supplementary material and will also be provided later with the extended version of the paper.

(2) in general. Obviously, to find an optimal threshold θ, access to the entire joint distribution is needed. However, this is not the main problem here, since in the next section, we will show that only a polynomial number of parameters of the joint distribution is needed. What is more interesting is the observation that the F-maximizer is in general not consistent with the order of marginal label probabilities. In fact, the regret can be substantial, as shown by the following result.

Proposition 2.3. *Let $\boldsymbol{h}_T$ be a vector of predictions obtained by putting a threshold on sorted marginal probabilities in the optimal way, then the worst-case regret is lower bounded by*

$$\sup_p \left(\mathbb{E}_{\mathbf{Y}}\left[F(\mathbf{Y}, \boldsymbol{h}_F^*) - F(\mathbf{Y}, \boldsymbol{h}_T)\right]\right) \geq \max(0, \frac{1}{6} - \frac{2}{m+4}),$$

where the supremum is taken over all possible distributions $p(\boldsymbol{Y})$.[3]

This is a rather surprising result in light of the existence of many algorithms that rely on finding a threshold for maximizing the F-measure [5, 9, 10]. While being justified by Theorems 2.1 and 2.3 for specific applications, this approach does not yield optimal predictions in general.

3 An Exact Algorithm for F-Measure Maximization

We now introduce an exact and efficient algorithm for computing the F-maximizer without using any additional assumption on the probability distribution $p(\boldsymbol{Y})$. While adopting the idea of decomposing the problem into an outer and an inner maximization, our algorithm differs from Jansche's in the way the inner maximization is solved. As a key element, we consider equivalence classes for the labels in terms of the number of ones in the vectors $\boldsymbol{h}$ and $\boldsymbol{y}$. The optimization of the F-measure can be substantially simplified by using these equivalence classes, since $\boldsymbol{h}$ and $\boldsymbol{y}$ then only appear in the numerator of the objective function. First, we show that only $m^2 + 1$ parameters of the joint distribution $p(\boldsymbol{Y})$ are needed to compute the F-maximizer.

Theorem 3.1. *Let $s_{\boldsymbol{y}} = \sum_{i=1}^m y_i$. The solution of (2) can be computed by solely using $p(\boldsymbol{Y} = \mathbf{0})$ and the values of*

$$p_{is} = p(Y_i = 1\,,\; s_{\boldsymbol{y}} = s), \qquad i, s \in \{1, \ldots, m\}\,,$$

which constitute an $m \times m$ matrix P.

Proof. The inner optimization problem (3) can be formulated as follows:

$$\boldsymbol{h}^{(k)^*} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})}\left[F(\boldsymbol{y}, \boldsymbol{h})\right] = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \sum_{\boldsymbol{y} \in \{0,1\}^m} p(\boldsymbol{y}) \frac{2\sum_{i=1}^m y_i h_i}{s_{\boldsymbol{y}} + k} \quad .$$

The sums can be swapped, resulting in

$$\boldsymbol{h}^{(k)^*} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \; 2\sum_{i=1}^m h_i \sum_{\boldsymbol{y} \in \{0,1\}^m} \frac{p(\boldsymbol{y}) y_i}{s_{\boldsymbol{y}} + k}. \tag{5}$$

Furthermore, one can sum up the probabilities $p(\boldsymbol{y})$ for all $\boldsymbol{y}$s with an equal value of $s_{\boldsymbol{y}}$. By using

$$p_{is} = \sum_{\boldsymbol{y} \in \{0,1\}^m : s_{\boldsymbol{y}} = s} y_i p(\boldsymbol{y})\,,$$

one can transform (5) into the following expression:

$$\boldsymbol{h}^{(k)^*} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \; 2\sum_{i=1}^m h_i \sum_{s=1}^m \frac{p_{is}}{s + k} \tag{6}$$

As a result, one does not need the whole distribution to solve (3), but only the values of p_{is}, which can be given in the form of an $m \times m$ matrix P with entries p_{is}. For the special case of $k = 0$, we have $\boldsymbol{h}^{(k)^*} = \mathbf{0}$ and $\mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})}\left[F(\boldsymbol{y}, \mathbf{0})\right] = p(\boldsymbol{Y} = \mathbf{0})$. □

[3] Finding the exact value of the supremum is an interesting open question.

Algorithm 1 General F-measure Maximizer

INPUT: matrix P and probability $p(\boldsymbol{Y} = \mathbf{0})$

define matrix W with elements given by Eq. 7;
compute $\mathrm{F} = \mathrm{PW}$
for $k = 1$ to m **do**
  solve the inner optimization problem (3) that can be reformulated as:

$$\boldsymbol{h}^{(k)^*} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \; 2 \sum_{i=1}^{m} h_i \mathrm{f}_{ik}$$

  by setting $h_i{=}1$ for top k elements in the k-th column of matrix F, and $h_i{=}0$ for the rest;
  store a value of

$$\mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})} \left[F(\boldsymbol{y}, \boldsymbol{h}^{(k)^*}) \right] = 2 \sum_{i=1}^{m} h_i^{(k)^*} \mathrm{f}_{ik};$$

end for
for $k = 0$ **take** $\boldsymbol{h}^{(k)^*} = \mathbf{0}$, and $\mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})} \left[F(\boldsymbol{y}, \mathbf{0}) \right] = p(\boldsymbol{Y} = \mathbf{0})$;
solve the outer optimization problem (4):

$$\boldsymbol{h}_F^* = \operatorname*{arg\,max}_{\boldsymbol{h} \in \{\boldsymbol{h}^{(0)^*}, \ldots, \boldsymbol{h}^{(m)^*}\}} \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})} \left[F(\boldsymbol{y}, \boldsymbol{h}) \right];$$

return $\boldsymbol{h}_F^*$ and $\mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{Y})} \left[F(\boldsymbol{y}, \boldsymbol{h}_F^*) \right]$;

If the matrix P is given, the solution of (2) is straight-forward. To simplify the notation, let us introduce an $m \times m$ matrix W with elements

$$w_{sk} = \frac{1}{s+k} \;\; , \qquad s, k \in \{1, \ldots, m\} \;\; , \tag{7}$$

The resulting algorithm, referred to as General F-measure Maximizer (GFM), is summarized in Algorithm 1 and its time complexity is analyzed in the following theorem.

Theorem 3.2. *Algorithm 1 solves problem (2) in time $o(m^3)$ assuming that the matrix* P *of m^2 parameters and $p(\boldsymbol{Y} = \mathbf{0})$ are given.*

Proof. We can notice in (6) that the sum $s + k$ assumes at most $m + 1$ values (it varies from s to $s + m$). By introducing the matrix W with elements (7), we can simplify (6) to

$$\boldsymbol{h}^{(k)^*} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \; 2 \sum_{i=1}^{m} h_i \mathrm{f}_{ik} \, , \tag{8}$$

where f_{ik} are elements of a matrix $\mathrm{F} = \mathrm{PW}$. To solve (8), it is enough to find the top k elements (i.e., the elements with the highest values) in the k-th column of matrix F, which can be carried out in linear time [15]. The solution of the outer optimization problem (4) is then straight-forward. Consequently, the complexity of the algorithm is dominated by a matrix multiplication that is solved naively in $\mathcal{O}(m^3)$, but faster algorithms working in $\mathcal{O}(m^{2.376})$ are known [16].[4] □

Let us briefly discuss the properties of our algorithm in comparison to the other algorithms discussed in Section 2. First of all, MEUF is characterized by a much higher time complexity being $\mathcal{O}(m^4)$ for the exact version. The recommended approximate variant reduces this complexity to $\mathcal{O}(m^3)$. In turn, the GFM algorithm has a complexity of $o(m^3)$. In addition, let us also remark that this complexity can be further decreased if the number of distinct values of $s_{\boldsymbol{y}}$ with non-zero probability mass is smaller than m.

Moreover, the MEUF framework will not deliver an exact F-maximizer if the assumption of independence is violated. On the other hand, MEUF relies on a smaller number of parameters (m values

[4] The complexity of the Coppersmith-Winograd algorithm [16] is more of theoretical significance, since practically this algorithm outperforms the naïve method only for huge matrices.

representing marginal probabilities). Our approach needs $m^2 + 1$ parameters, but then computes the maximizer exactly. Since estimating a larger number of parameters is statistically more difficult, it is a priori unclear which method performs better in practice.

Our algorithm can also be tailored for finding an optimal threshold. It is then simplified due to constraining the number of hypotheses. Instead of finding the top k elements in the k-th column, it is enough to rely on the order of the marginal probabilities $p_i = \sum_{s=1}^{m} p_{is}$. As a result, there is no need to compute the entire matrix F; instead, only the elements that correspond to the k highest marginal probabilities for each column k are needed. Of course, the thresholding can be further simplified by verifying only a small number $t < m$ of thresholds.

4 Application of the Algorithm

The GFM algorithm can be used whenever an estimation of the distribution $p(\boldsymbol{Y})$ or, alternatively, estimates of the matrix P and probability $p(\boldsymbol{Y} = \boldsymbol{0})$ are available. In this section, we focus on the application of GFM in the multi-label setting. Thus, we consider the task of predicting a vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0, 1\}^m$ given another vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ as input attributes. To this end, we train a classifier $\boldsymbol{h}(\boldsymbol{x})$ on a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ and perform inference for a given test vector $\boldsymbol{x}$ so as to deliver an optimal prediction under the F-measure (1). Thus, we optimize the performance for each instance individually (instance-wise F-measure), in contrast to macro- and micro-averaging of the F-measure.

We follow an approach similar to Conditional Random Fields (CRFs) [17, 18], which estimates the joint conditional distribution $p(\boldsymbol{Y} \mid \boldsymbol{x})$. This approach has the additional advantage that one can easily sample from the estimated distribution. The underlying idea is to repeatedly apply the product rule of probability to the joint distribution of the labels $\boldsymbol{Y} = (Y_1, \ldots, Y_m)$:

$$p(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{x}) = \prod_{k=1}^{m} p(Y_k = y_k \mid \boldsymbol{x}, y_1, \ldots, y_{k-1}) \tag{9}$$

This approach, referred to as Probabilistic Classifier Chains (PCC), has proved to yield state-of-the-art performance in MLC [19]. Learning in this framework can be considered as a procedure that relies on constructing probabilistic classifiers for estimating $p(Y_k = y_k | \boldsymbol{x}, y_1, \ldots, y_{k-1})$, independently for each $k = 1, \ldots, m$. To sample from the conditional joint distribution $p(\boldsymbol{Y} \mid \boldsymbol{x})$, one follows the chain and picks the value of label y_k by tossing a biased coin with probabilities given by the k-th classifier. Based on a sample of observations generated in this way, our GFM algorithm can be used to perform the optimal inference under F-measure.

In the experiments, we train PCC by using linear regularized logistic regression. By plugging the log-linear model into (9), it can be shown that pairwise dependencies between labels y_i and y_j can be modeled. We tune the regularization parameter using 3-fold cross-validation. To perform inference, we draw for each test example a sample of 200 observations from the estimated conditional distribution. We then apply five inference methods. The first one (H) estimates marginal probabilities $p_i(\boldsymbol{x})$ and predicts 1 for labels with $\hat{p}_i(\boldsymbol{x}) \geq 0.5$; this is an optimal strategy for the Hamming loss. The second method (MEUF) uses the estimates $\hat{p}_i(\boldsymbol{x})$ for computing the F-measure by applying the MEUF method. If the labels are independent, this method computes the F-maximizer exactly. As a third method, we use the approximate cubic-time variant of MEUF with the parameters suggested in the original paper [12]. Finally, we use GFM and its variant that finds the optimal threshold (GFM-T).

Before showing the results of PCC on benchmark datasets, let us discuss results for two synthetic models, one with independent and another one with dependent labels. Plots and a description of the models are given in Fig. 1. As can be observed, MEUF performs the best for independent labels, while GFM approaches its performance if the sample size increases. This is coherent with our theoretical analysis, since GFM needs to estimate more parameters. However, in the case of dependent labels, MEUF performs poorly, even for a larger sample size, since the underlying assumption is not satisfied. Interestingly, both approximate variants perform very similarly to the original algorithms. We also see that GFM has a huge advantage over MEUF regarding the time complexity.[5]

[5] All the computations are performed on a typical desktop machine.

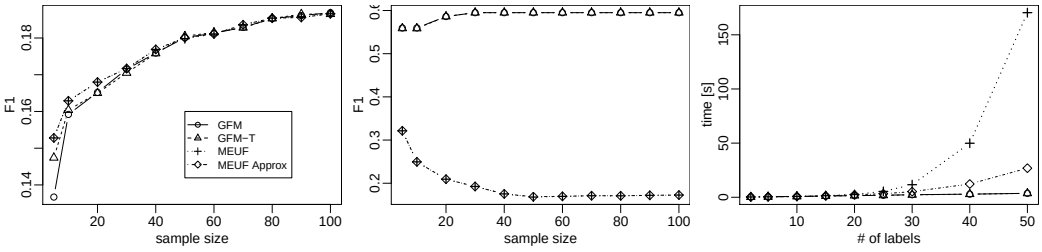

Figure 1: The plots show the performance under the F-measure of the inference methods: GFM, its thresholding variant GFM-T, MEUF, and its approximate version MEUF Approx. Left: the performance as a function of sample size generated from independent distribution with $p_i = 0.12$ and $m = 25$ labels. Center: similarly as above, but the distribution is defined according to (9), where all $p(Y_i = y_i \mid y_1, \ldots, y_{i-1})$ are defined by logistic models with a linear part $-\frac{1}{2}(i-1)+\sum_{j=1}^{i-1} y_j$. Right: running times as a function of the number of labels with a sample size of 200. All the results are averaged over 50 trials.

Table 1: Experimental results on four benchmark datasets. For each dataset, we give the number of labels (m) and the size of training and test sets (in parentheses: training/test set). A "-" symbol indicates that an algorithm did not complete the computations in a reasonable amount of time (several days). In bold: the best results for a given dataset and performance measure.

METHOD	HAMMING LOSS	MACRO-F	MICRO-F	F	INFERENCE TIME [S]	HAMMING LOSS	MACRO-F	MICRO-F	F	INFERENCE TIME [S]
	SCENE: $m = 6$ (1211/1169)					YEAST: $m = 14$ (1500/917)				
PCC H	0.1030	0.6673	0.6675	0.5779	0.969	0.2046	0.3633	0.6391	0.6160	3.704
PCC GFM	0.1341	**0.7159**	0.6915	**0.7101**	0.985	0.2322	0.4034	0.6554	0.6479	3.796
PCC GFM-T	0.1343	0.7154	0.6908	0.7094	1.031	0.2324	0.4039	0.6553	0.6476	3.907
PCC MEUF APPROX.	0.1323	0.7131	0.6910	0.6977	1.406	0.2295	0.4030	0.6551	0.6469	10.000
PCC MEUF	0.1323	0.7131	0.6910	0.6977	1.297	0.2292	0.4034	0.6557	0.6477	11.453
BR	**0.1023**	0.6591	0.6602	0.5542	1.125	**0.1987**	0.3349	0.6299	0.6039	0.640
BR MEUF APPROX.	0.1140	0.7048	**0.6948**	0.6468	1.579	0.2248	**0.4098**	**0.6601**	**0.6527**	7.110
BR MEUF	0.1140	0.7048	0.6948	0.6468	2.094	0.2263	0.4096	0.6591	0.6523	10.031
	ENRON: $m = 53$ (1123/579)					MEDIAMILL: $m = 101$ (30999/12914)				
PCC H	0.0471	0.1141	0.5185	0.4892	195.061	**0.0304**	0.0931	0.5577	0.5429	1405.772
PCC GFM	0.0521	0.1618	0.5943	0.6006	194.889	0.0348	0.1491	0.5849	0.5734	1420.663
PCC GFM-T	0.0521	**0.1619**	0.5948	**0.6011**	196.030	0.0348	0.1499	0.5854	0.5737	1464.147
PCC MEUF APPROX.	0.0523	0.1612	0.5932	0.6007	1081.837	0.0350	0.1504	0.5871	0.5740	308582.019
PCC MEUF	0.0523	0.1612	0.5932	0.6007	6676.145	-	-	-	-	-
BR	**0.0468**	0.1049	0.5223	0.4821	8.594	**0.0304**	0.1429	0.5623	0.5462	207.655
BR MEUF APPROX.	0.0513	0.1554	**0.5969**	0.5947	850.494	0.3508	**0.1917**	**0.5889**	**0.5744**	258431.125
BR MEUF	0.0513	0.1554	0.5969	0.5947	7014.453	-	-	-	-	-

The results on four commonly used benchmark datasets[6] with known training and test sets are presented in Table 1, which also includes some basic statistics of these datasets. We additionally present results of the binary relevance (BR) approach which trains an independent classifier for each label (we used the same base learner as in PCC). We also apply the MEUF method on marginals delivered by BR. This is the best we can do if only marginals are known. From the results of the F-measure, we can clearly state that all approaches tailored for this measure obtain better results. However, there is no clear winner among them. It seems that in practical applications, the theoretical results concerning the worst-case scenario do not directly apply. Also, the number of parameters to be estimated does not play an important role. However, GFM drastically outperforms MEUF in terms of computational complexity. For the Mediamill dataset, the MEUF algorithm in its exact version did not complete the computations in a reasonable amount of time. The running times for the approximate version are already unacceptably high for this dataset.

We also report results for the Hamming loss, macro- and micro-averaging F-measure. We can see, for example, that approaches appropriate for Hamming loss obtain the best results regarding this measure. The macro and micro F-measure are presented mainly as a reference. The former is computed by averaging the F-measure label-wise, while the latter concatenates all test examples and computes a single value over all predictions. These two variants of the F-measure are not directly optimized by the algorithms used in the experiment.

[6]These datasets are taken from the MULAN (http://mulan.sourceforge.net/datasets.html) and LibSVM (http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multilabel.html) repositories.

5 Discussion

The GFM algorithm can be considered for maximizing the macro F-measure, for example, in a similar setting as in [10], where a specific Bayesian on-line model is used. In order to maximize the macro F-measure, the authors sample from the graphical model to find an optimal threshold. The GFM algorithm may solve this problem optimally, since, as stated by the authors, the independence of labels is lost after integrating out the model parameters. Theoretically, one may also consider a direct maximization of the micro F-measure with GFM, but the computational burden is rather high in this case.

Interestingly, there are no other MLC algorithms that maximize the F-measure in an instance-wise manner. We also cannot refer to other results already published in the literature, since usually only the micro- and macro-averaged F-measures are reported [20, 11]. This is rather surprising, especially since some closely related measures are often computed in the instance-wise manner in empirical studies. For example, the Jaccard distance (sometimes referred to as accuracy [21]), which differs from the F-measure in an additional term in the denominator, is commonly used in such a way.

The situation is slightly different in structured output prediction, where algorithms for instance-wise maximization of the F-measure do exist. These include, for example, struct SVM [6], SEARN [8], and a specific variant of CRFs [7]. Usually, these algorithms are based on additional assumptions, like label independence in struct SVM. The GFM algorithm can also be easily tailored for maximizing the instance-wise F-measure in structured output prediction, in a similar way as presented above. If the structured output classifier is able to model the joint distribution from which we can easily sample observations, then the use of the algorithm is straight-forward. An application of this kind is planned as future work.

Surprisingly, in both papers [8] and [6], experimental results are reported in terms of micro F-measure, although the algorithms maximize the instance-wise F-measure on the training set. Needless to say, one should not expect such an approach to result in optimal performance for the micro-averaged F-measure. Despite being related to each other, these two measures coincide only in the specific case where $\sum_{i=1}^{m}(y_i + h_i)$ is constant for all test examples. The discrepancy between these measures strongly depends on the nature of the data and the classifier used. For high variability in $\sum_{i=1}^{m}(y_i + h_i)$, a significant difference between the values of these two measures is to be expected.

The use of the GFM algorithm in binary classification seems to be superfluous, since in this case, the assumption of label independence is rather reasonable. MEUF seems to be the right choice for probabilistic classifiers, unless its application is prevented due to its computational complexity. Thresholding methods [5] or learning algorithms optimizing the F-measure directly [2, 3, 4] are probably the most appropriate solutions here.

6 Conclusions

In contrast to other performance measures commonly used in experimental studies, such as misclassification error rate, squared loss, and AUC, the F-measure has been investigated less thoroughly from a theoretical point of view so far. In this paper, we analyzed the problem of optimal predictive inference from the joint distribution under the F-measure. While partial results were already known from the literature, we completed the picture by presenting the solution for the general case without any distributional assumptions. Our GFM algorithm requires only a polynomial number of parameters of the joint distribution and delivers the exact solution in polynomial time. From a theoretical perspective, GFM should be preferred to existing approaches, which typically perform threshold maximization on marginal probabilities, often relying on the assumption of (conditional) independence of labels.

Acknowledgments. Krzysztof Dembczyński has started this work during his post-doctoral stay at Philipps-Universität Marburg supported by German Research Foundation (DFG) and finalized it at Poznań University of Technology under the grant 91-515/DS of the Polish Ministry of Science and Higher Education. Willem Waegeman is supported as a postdoc by the Research Foundation of Flanders (FWO-Vlaanderen). The part of this work has been done during his visit at Philipps-Universität Marburg. Weiwei Cheng and Eyke Hüllermeier are supported by DFG. We also thank the anonymous reviewers for their valuable comments.

References

[1] C. J. van Rijsbergen. Foundation of evaluation. *Journal of Documentation*, 30(4):365–373, 1974.

[2] David R. Musicant, Vipin Kumar, and Aysel Ozgur. Optimizing F-measure with support vector machines. In *FLAIRS-16, 2003*, pages 356–360, 2003.

[3] Thorsten Joachims. A support vector method for multivariate performance measures. In *ICML 2005*, pages 377–384, 2005.

[4] Martin Jansche. Maximum expected F-measure training of logistic regression models. In *HLT/EMNLP 2005*, pages 736–743, 2005.

[5] Sathiya Keerthi, Vikas Sindhwani, and Olivier Chapelle. An efficient method for gradient-based adaptation of hyperparameters in SVM models. In *Advances in Neural Information Processing Systems 19*, 2007.

[6] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

[7] Jun Suzuki, Erik McDermott, and Hideki Isozaki. Training conditional random fields with multivariate evaluation measures. In *ACL*, pages 217–224, 2006.

[8] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009.

[9] Rong-En Fan and Chih-Jen Lin. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University, 2007.

[10] Xinhua Zhang, Thore Graepel, and Ralf Herbrich. Bayesian online learning for multi-label and multi-variate performance measures. In *AISTATS 2010*, pages 956–963, 2010.

[11] James Petterson and Tiberio Caetano. Reverse multi-label learning. In *Advances in Neural Information Processing Systems 23*, pages 1912–1920, 2010.

[12] Martin Jansche. A maximum expected utility framework for binary sequence labeling. In *ACL 2007*, pages 736–743, 2007.

[13] David Lewis. Evaluating and optimizing autonomous text classification systems. In *SIGIR 1995*, pages 246–254, 1995.

[14] Juan Jose del Coz, Jorge Diez, and Antonio Bahamonde. Learning nondeterministic classifiers. *J. Mach. Learn. Res.*, 10:2273–2293, 2009.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 2nd edition*. MIT Press, 2001.

[16] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 3(9):251–280, 1990.

[17] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pages 282–289, 2001.

[18] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *CIKM 2005*, pages 195–200, 2005.

[19] Krzysztof Dembczyński, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML 2010*, pages 279–286, 2010.

[20] Piyush Rai and Hal Daumé III. Multi-label prediction via sparse infinite CCA. In *Advances in Neural Information Processing Systems 22*, pages 1518–1526, 2009.

[21] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.