Optimizing the F-Measure in Multi-Label Classification: Plug-in Rule Approach versus Structured Loss Minimization

Krzysztof Dembczyński[1] KDEMBCZYNSKI@CS.PUT.POZNAN.PL
Arkadiusz Jachnik[1] AJACHNIK@CS.PUT.POZNAN.PL
Wojciech Kotłowski[1] WKOTLOWSKI@CS.PUT.POZNAN.PL
Willem Waegeman[2] WILLEMWAEGEMAN@GMAIL.COM
Eyke Hüllermeier[3] EYKE@MATHEMATIK.UNI-MARBURG.DE
[1]Institute of Computing Science, Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland
[2]NGDATA-Europe, Dok-Noord 7, 9000 Gent, Belgium
[3]Mathematics and Computer Science, Marburg University, Hans-Meerwein-Str., 35032 Marburg, Germany

Abstract

We compare the plug-in rule approach for optimizing the F_β-measure in multi-label classification with an approach based on structured loss minimization, such as the structured support vector machine (SSVM). Whereas the former derives an optimal prediction from a probabilistic model in a separate inference step, the latter seeks to optimize the F_β-measure directly during the training phase. We introduce a novel plug-in rule algorithm that estimates all parameters required for a Bayes-optimal prediction via a set of multinomial regression models, and we compare this algorithm with SSVMs in terms of computational complexity and statistical consistency. As a main theoretical result, we show that our plug-in rule algorithm is consistent, whereas the SSVM approaches are not. Finally, we present results of a large experimental study showing the benefits of the introduced algorithm.

1. Introduction

Motivated by applications such as tag recommendation in computer vision or gene function prediction in bioinformatics, the machine learning community has witnessed a rapid expansion of research on so-called multi-label classification (MLC) in recent years. MLC can be seen as specific type of structured output prediction, and also shares commonalities with multi-task learning. The generalization from predicting a single binary label, like in conventional classification, to predicting a vector of such labels gives rise to a number of theoretical challenges; this includes the possibility to model statistical dependencies between different class labels as well as to define and minimize appropriate loss functions.

A large number of MLC loss functions has been proposed in the literature. In experimental studies, many of these losses are typically analyzed simultaneously. Yet, it is clear that a method performing optimally for one loss is likely to perform suboptimally for another loss. There are, however, a few general frameworks that can indeed be tailored for different loss functions in a generic way—an important example of such a framework is structured support vector machines (SSVMs).

For simple loss functions, analytic expressions of the Bayes (optimal) classifier can be derived. For example, it is known that the Hamming loss minimizer coincides with the marginal modes of the conditional distribution of the class labels given an instance, and methods such as binary relevance, stacking or M3L (Hariharan et al., 2012) perform particularly well in this case. Conversely, for the subset 0/1 loss, the risk minimizer is given by the joint mode of the conditional distribution, for which methods such as the label powerset classifier, conditional random fields and SSVMs without margin rescaling might be good choices.

For complex multi-label loss functions, the picture becomes more blurry, and the minimization of such losses requires more involved procedures. The F_β-measure is a specifically interesting example. Despite being

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

encountered in many application domains, algorithms suitable for optimizing this measure have been introduced only recently. They can be subdivided in two categories. Structured loss minimization approaches such as (Petterson & Caetano, 2010; 2011) intend to maximize the F_β-measure during the training phase in frameworks like SSVMs, whereas plug-in rule approaches (or decision-theoretic approaches) such as (Lewis, 1995; Chai, 2005; Jansche, 2007; Dembczyński et al., 2011; Quevedo et al., 2012; Ye et al., 2012) deduce F_β-measure maximizing predictions from a probabilistic model during an inference step. Let us notice that a similar distinction is considered by Ye et al. (2012); however, while our focus is on multi-label problems, their analysis is more relevant for binary classification.

As shown by Dembczyński et al. (2011), $m^2 + 1$ parameters of the conditional joint distribution (with m the number of labels) are needed to obtain the Bayes classifier for the latter type of methods. Departing from those results, we propose a novel algorithm that estimates these parameters directly via a reduction to a set of multinomial regression problems. Compared to the approach of Dembczyński et al. (2011) that constructs a model for the conditional joint distribution, our method avoids an approximation during the inference phase due to sampling from the probabilistic model. In addition, we analyze and compare the plug-in rule and structured loss minimization methods in terms of computational complexity and statistical consistency. Our main theoretical results show that our algorithm is consistent, while the SSVM approach followed in (Petterson & Caetano, 2010; 2011) is not. Finally, we present results of a large experimental study, in which we thoroughly compare both approaches.

2. Multi-Label Classification

We start with a more formal definition of the MLC problem. Let $\mathcal{X}$ denote an instance space, and let $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ be a finite set of class labels. We assume that an instance $\boldsymbol{x} \in \mathcal{X}$ is (non-deterministically) associated with a subset of labels $L \in 2^{\mathcal{L}}$; this subset is often called the set of relevant (positive) labels, while the complement $\mathcal{L} \setminus L$ is considered as irrelevant (negative) for $\boldsymbol{x}$. We identify a set L of relevant labels with a binary vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$, in which $y_i = 1$ iff $\lambda_i \in L$. The set of possible labelings is denoted $\mathcal{Y} = \{0, 1\}^m$. We assume observations to be generated independently and randomly according to a probability distribution $P(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y})$ (later denoted $P(\boldsymbol{x}, \boldsymbol{y})$) on $\mathcal{X} \times \mathcal{Y}$, i.e., an observation $(\boldsymbol{x}, \boldsymbol{y})$ is the realization of two random vectors, $\boldsymbol{X} = (X_1, X_2, \ldots, X_q)$ and $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_m)$.

A multilabel classifier $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))$ assigns a (predicted) label subset to each instance $\boldsymbol{x} \in \mathcal{X}$. The prediction accuracy of $\boldsymbol{h}$ is measured in terms of its *risk*, that is, its expected loss

$$L(\boldsymbol{h}, P) = \mathbb{E}\left[\ell(\boldsymbol{Y}, \boldsymbol{h}(\boldsymbol{X}))\right] = \int \ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\, dP(\boldsymbol{x}, \boldsymbol{y})\,, \quad (1)$$

where $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a *loss function*. In addition, it will be convenient to use an expected loss conditioned on an instance $\boldsymbol{x} \in \mathcal{X}$:

$$L(\boldsymbol{h}, P \,|\, \boldsymbol{x}) = \mathbb{E}\left[\ell(\boldsymbol{Y}, \boldsymbol{h}(\boldsymbol{x})) \,|\, \boldsymbol{x}\right] = \sum_{\boldsymbol{y} \in \mathcal{Y}} \ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) P(\boldsymbol{y} \,|\, \boldsymbol{x})\,,$$

so that $L(\boldsymbol{h}, P) = \mathbb{E}[L(\boldsymbol{h}, P \,|\, \boldsymbol{X})]$.

The optimal classifier, commonly referred to as *Bayes classifier*, minimizes the risk conditioned on $\boldsymbol{x}$:

$$\boldsymbol{h}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{h} \in \mathcal{Y}} \sum_{\boldsymbol{y} \in \mathcal{Y}} \ell(\boldsymbol{y}, \boldsymbol{h}) P(\boldsymbol{y} \,|\, \boldsymbol{x})\,. \quad (2)$$

We note that $\boldsymbol{h}^*$ is in general not unique. However, the risk of $\boldsymbol{h}^*$, denoted $L^*(P)$, is unique, and is called the *Bayes risk*.

Label subsets can be compared in terms of a multitude of loss functions, many of which lead to intractable optimization problems. Therefore, the actual loss function, often referred to as the *task loss*, is usually replaced by a *surrogate loss* that is easier to cope with (e.g., a convex upper bound of the task loss). Alternatively, the original loss minimization problem can be decomposed into problems of simpler type, for which efficient algorithmic solutions are readily available.

3. The F_β-Measure

Given a prediction $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x})) \in \mathcal{Y}$ of a binary label vector $\boldsymbol{y} = (y_1, \ldots, y_m)$, the F_β-measure is defined as follows:

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

where $0/0 = 1$ by definition. This measure essentially corresponds to the weighted harmonic mean of precision and recall. Compared to measures like Hamming loss, the F_β-measure enforces a better balance between performance on relevant and irrelevant labels, and, therefore, it is more suitable for multi-label problems, in which irrelevant labels often prevail.

Since the F_β-measure is a score of adjustment between $\boldsymbol{y}$ and $\boldsymbol{h}$, it is actually not a loss function but rather

a kind of utility measure. Therefore, we will either consider the F_β-based loss function (F_β-loss for short)

$$\ell_{F_\beta}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = 1 - F_\beta(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\,, \tag{3}$$

or speak of utility maximization instead of risk minimization.

4. Bayes Classifier for the F_β-Measure

Unfortunately, there is no closed-form of the Bayes classifier that maximizes the F_β-measure. However, as shown by Dembczyński et al. (2011), the Bayes classification can be computed efficiently, even in the general case without any assumption about the underlying probability distribution.

The Bayes classifier for the F_β-measure is given by (for the sake of clarity, we will suppress dependence on $\boldsymbol{x}$ in the notation, whenever it is clear from the context):

$$\begin{aligned}\boldsymbol{h}^* &= \arg\max_{\boldsymbol{h}\in\mathcal{Y}} \mathbb{E}\left[F_\beta(\boldsymbol{Y}, \boldsymbol{h})\right] = \arg\max_{\boldsymbol{h}\in\mathcal{Y}} \sum_{\boldsymbol{y}\in\mathcal{Y}} F_\beta(\boldsymbol{y}, \boldsymbol{h}) P(\boldsymbol{y}) \\ &= \arg\max_{\boldsymbol{h}\in\mathcal{Y}} \sum_{\boldsymbol{y}\in\mathcal{Y}} \frac{(1+\beta^2)\sum_{i=1}^m y_i h_i P(\boldsymbol{y})}{\beta^2 \sum_{i=1}^m y_i + \sum_{i=1}^m h_i}\,. \end{aligned} \tag{4}$$

Problem (4) can be solved via outer and inner maximization (Jansche, 2007). Namely, it can be transformed into an inner maximization

$$\boldsymbol{h}_k^* = \arg\max_{\boldsymbol{h}\in\mathcal{H}_k} \mathbb{E}\left[F_\beta(\boldsymbol{Y}, \boldsymbol{h})\right]\,, \tag{5}$$

where $\mathcal{H}_k = \{\boldsymbol{h} \in \mathcal{Y} \mid \sum_{i=1}^m h_i = k\}$, followed by an outer maximization

$$\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}\in\{\boldsymbol{h}_0^*,\ldots,\boldsymbol{h}_m^*\}} \mathbb{E}\left[F_\beta(\boldsymbol{Y}, \boldsymbol{h})\right]\,. \tag{6}$$

The outer maximization (6) can be done by simply checking all $m+1$ possibilities. The main effort is then required for solving the inner maximization (5).

Let us consider the inner maximization problem for a given $k > 0$. We first introduce the following notation:

$$s_{\boldsymbol{y}} = \sum_{i=1}^m y_i, \quad \Delta_{ik}^u = \sum_{\boldsymbol{y}: y_i = u} \frac{P(\boldsymbol{y})}{\beta^2 s_{\boldsymbol{y}} + k}\,.$$

Then, we can write

$$\boldsymbol{h}_k^* = \arg\max_{\boldsymbol{h}\in\mathcal{H}_k} \sum_{\boldsymbol{y}\in\mathcal{Y}} \frac{(1+\beta^2)\sum_{i=1}^m y_i h_i P(\boldsymbol{y})}{\beta^2 s_{\boldsymbol{y}} + k}\,. \tag{7}$$

By swapping sums in (7), we get

$$\begin{aligned}\boldsymbol{h}_k^* &= \arg\max_{\boldsymbol{h}\in\mathcal{H}_k} (1+\beta^2) \sum_{i=1}^m h_i \sum_{\boldsymbol{y}\in\mathcal{Y}} \frac{y_i P(\boldsymbol{y})}{\beta^2 s_{\boldsymbol{y}} + k} \\ &= \arg\max_{\boldsymbol{h}\in\mathcal{H}_k} (1+\beta^2) \sum_{i=1}^m h_i \Delta_{ik}^1\,. \end{aligned} \tag{8}$$

Since there are only k labels for which we can set $h_i = 1$, we obtain the optimal solution of the inner maximization by setting $h_i = 1$ for the top k values of Δ_{ik}. For the solution $\boldsymbol{h}_k^*$, we compute $\mathbb{E}\left[F_\beta(\boldsymbol{Y}, \boldsymbol{h}_k^*)\right]$, which is further used in the outer maximization. For the specific case of $\boldsymbol{h}_0 = \boldsymbol{0}$, $\mathbb{E}\left[F_\beta(\boldsymbol{Y}, \boldsymbol{h}_0)\right] = P(\boldsymbol{Y} = \boldsymbol{0})$.

Thus, we only need Δ_{ik}^1 for $1 \le i, k \le m$ and $P(\boldsymbol{0})$, that is, $m^2 + 1$ parameters to compute the optimal prediction. With these parameters, the solution can be obtained in $\mathcal{O}(m^2)$ time, i.e., the dominating part of the procedure is the inner maximization: For each k, a selection of the top k elements must be done, which can be accomplished in linear time. We will call this approach the *General F_β-Measure Maximizer* (GFM).

Under the assumption of independence of the labels $Y_1, \ldots, Y_m$, the optimization problem (4) can be substantially simplified. Lewis (1995) and Jansche (2007) have shown independently that the optimal solution is either empty ($\boldsymbol{h}^* = \boldsymbol{0}$) or consists of those labels with the highest marginal probabilities $p_i = P(Y_i = 1)$. As a consequence, the form of the solution of the k-th inner maximization problem is known (the k labels with the highest marginal probabilities). The only missing element is to compute the value of the expected F_β-measure for a given k in order to select the best k.

Different approaches have been proposed to tackle this problem. Jansche (2007) introduced an algorithm that works in $\mathcal{O}(m^4)$ time. Chai (2005) and Quevedo et al. (2012) have independently derived $\mathcal{O}(m^3)$ algorithms based on dynamic programming. In a more recent follow-up paper, Ye et al. (2012) further improves the dynamic programming algorithm of Chai (2005) to an $\mathcal{O}(m^2)$ complexity for rational β by additional sharing of internal representations.

5. Plug-in Rule Classifier

One way to construct a classifier using training data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ is to estimate all required parameters and then plug them into the form of the Bayes classifier. Such an approach is usually referred to as *plug-in rule classifier*. In the following, we show that all parameters required by the Bayes classifier for the F_β-measure can be efficiently obtained thanks to a specific reduction to a set of multinomial regression problems.

Let us start with the estimation of the Δ_{ik}^1. Unfortunately, these quantities do not correspond to a proper probability distribution, i.e, Δ_{ik}^0 and Δ_{ik}^1 do not sum up to 1 or to any known constant; moreover, there is no other subset of these quantities either that may posses a property of that kind. Yet, there is a simple trick that we can use. Let us denote by $\boldsymbol{P}$ and $\boldsymbol{W}$ two

$m \times m$ matrices with elements

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s, \,|\, \boldsymbol{x}), \quad w_{ik} = (\beta^2 s + k)^{-1} ,$$

respectively. Then, the $m \times m$ matrix $\boldsymbol{\Delta}$ with elements Δ_{ik} can be obtained by

$$\boldsymbol{\Delta} = \boldsymbol{P}\boldsymbol{W} .$$

The matrix $\boldsymbol{P}$ can be estimated by using a simple reduction, namely by solving m multi-class probability estimation problems (e.g., multinomial regression) with at most $m + 1$ classes. The scheme of the reduction for the i-th subproblem is the following:

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![y_i = 1]\!] \cdot s_{\boldsymbol{y}}) .$$

For a given $\boldsymbol{x}$, we estimate $P(y = [\![y_i = 1]\!] \cdot s_{\boldsymbol{y}} \,|\, \boldsymbol{x})$, $y \in \{0, \ldots, m\}$. We can model the probabilities $P(0 \,|\, \boldsymbol{x}), \ldots, P(m \,|\, \boldsymbol{x})$ with a function $f(y, \boldsymbol{x})$ using the multinomial logistic transform:

$$P(y \,|\, \boldsymbol{x}) = \frac{\exp(f(y, \boldsymbol{x}))}{\sum_{j=0}^{m} \exp(f(j, \boldsymbol{x}))}$$

Then, the logistic loss has the form

$$\ell_{\log}(y, \boldsymbol{x}, f) = \log \left(\sum_{j=0}^{m} \exp(f(j, \boldsymbol{x})) \right) - f(y, \boldsymbol{x}) .$$

The learning can be formulated as regularized minimization of the logistic loss:

$$f^*(y, \boldsymbol{x}) = \arg\min_f \frac{1}{n} \sum_{i=1}^{n} \ell_{\log}(y, \boldsymbol{x}, f) + \lambda J(f) ,$$

where λ controls the trade-off between the average loss over the training examples and the regularizer J that penalizes complex solutions. In a similar way, we can estimate $P(\boldsymbol{0} \,|\, \boldsymbol{x})$ by performing an additional reduction to the binary problem

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![\boldsymbol{y} = \boldsymbol{0}]\!]) ,$$

and solving it via logistic loss minimization.

The decomposition of the original problem into independent multinomial regression tasks has computational advantages. Moreover, since the number of distinct values of $s_{\boldsymbol{y}}$ is usually small, the number of classes in a single multinomial regression task is much smaller than $m + 1$; only in the worst case, we end up with a quadratic complexity in the number of labels m. Let us remark, however, that the quantities to be estimated across different tasks are not fully independent of each other (e.g., p_{im} is the same for all i). Consequently, learning on a finite training set may lead to conflicting estimates of $P(\boldsymbol{0} \,|\, \boldsymbol{x})$ and the matrix $\boldsymbol{P}$, that is, estimates that are not in agreement with any valid distribution. To avoid such conflicts, one may include additional constraints in the learning problem or calibrate the estimates afterwards. Another alternative is to train a model that estimates the conditional joint distribution, for example, by using probabilistic classifier chains; then, estimates of the required parameters can be obtained by sampling from this distribution. This approach has been adopted by Dembczyński et al. (2011). One should note, however, that sampling usually dominates the complexity of the inference.

To summarize, the procedure for learning a probabilistic model has a time complexity that is at most quadratic in m. In the inference phase, we first obtain estimates of $P(\boldsymbol{0} \,|\, \boldsymbol{x})$ and $\boldsymbol{P}$ for each test instance $\boldsymbol{x}$, again in at most quadratic time. The matrices $\boldsymbol{P}$ and $\boldsymbol{W}$ are multiplied to get $\boldsymbol{\Delta}$ in at most cubic time. Finally, all parameters are plugged into the GFM procedure, which requires quadratic time in m. For a moderate number of labels (up to hundreds) and a small number of distinct values of $s_{\boldsymbol{y}}$, this yields a feasible approach for MLC with F_β as performance measure. We refer to this method as *Exact-F_β-Plug-in classifier* (EFP).

Under the assumption of label independence, the plug-in rule approach simplifies a lot. Since we only need the marginal probabilities $p_i = P(Y_i = 1)$, we reduce the problem to m binary classification tasks. For label λ_i, the reduction takes the form

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = y_i) ,$$

and we can learn a classifier in a similar way as above, namely through minimization of the logistic loss. Then, for each test instance $\boldsymbol{x}$, we obtain a vector of marginal probabilities p_i, to which we apply, for example, the method of Ye et al. (2012). This results in a procedure that is much faster than the exact one. The learning is linear in m, and the inference is quadratic for rational β or cubic in the general case. We refer to this method as *Label-independence-F_β-Plug-in classifier* (LFP). This method, however, may lead to suboptimal results if the assumption of label independence is violated. Theoretically, the difference between these two approaches can become arbitrarily large (Dembczyński et al., 2011).

6. Structured Loss Minimization

An alternative to the plug-in rule approach outlined above is to minimize the task loss directly. Since

this is intractable for the F_β-loss, one usually minimizes a convex upper bound. Here, we examine the general framework of structured hinge loss minimization (Tsochantaridis et al., 2005), mainly following the approach of Petterson & Caetano (2010; 2011).

The problem can be stated as learning a function $f(\boldsymbol{y}, \boldsymbol{x})$ such that a prediction $\boldsymbol{h}(\boldsymbol{x})$ is given by:

$$\boldsymbol{h}(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} f(\boldsymbol{y}, \boldsymbol{x})\,. \tag{9}$$

The training consists of minimizing the structural hinge loss, which can be defined by

$$\tilde{\ell}_h(\boldsymbol{y}, \boldsymbol{x}, f) = \max_{\boldsymbol{y}'\in\mathcal{Y}}\{\ell(\boldsymbol{y}, \boldsymbol{y}') + f(\boldsymbol{y}', \boldsymbol{x})\} - f(\boldsymbol{y}, \boldsymbol{x})\,, \tag{10}$$

where $\ell(\boldsymbol{y}, \boldsymbol{y}')$ is the corresponding task loss. Here, we consider the F_β-loss ℓ_{F_β} (3). The learning problem can then be stated as

$$f^*(\boldsymbol{y}, \boldsymbol{x}) = \arg\min_f \frac{1}{n}\sum_{i=1}^{n} \tilde{\ell}_h(\boldsymbol{y}_i, \boldsymbol{x}_i, f) + \lambda J(f)\,,$$

where we trade-off the average hinge loss over the training examples and the regularization, similarly as for the logistic loss minimization in the plug-in rule approach. This is the so-called margin-rescaling estimator for SSVMs. The formulation leads to a quadratic program with exponentially many constraints. Therefore, one usually uses the cutting-plane algorithm (Kelley, 1960), which starts by solving the problem with no constraints and iteratively adds the most violated constraint for the current solution of the optimization problem. In each iteration, one thus needs to find

$$\boldsymbol{y}_i^* = \arg\max_{\boldsymbol{y}'\in\mathcal{Y}}\{\ell_{F_\beta}(\boldsymbol{y}_i, \boldsymbol{y}') + f(\boldsymbol{y}', \boldsymbol{x})\}\,. \tag{11}$$

Depending on the choice of $f(\boldsymbol{y}, \boldsymbol{x})$, one ends up with procedures of different complexity. Petterson & Caetano (2010) assume f to be decomposable over labels:

$$f(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}) = \sum_{i=1}^{m} f_i(y_i, \boldsymbol{x})\,. \tag{12}$$

This form of the function f leads to an effective formulation of the problem. It turns out that the constraint generation problem (11) can be solved in $\mathcal{O}(m^2)$ time. Moreover, the prediction problem (9) can be solved in linear time with respect to m:

$$h_i(\boldsymbol{x}) = \arg\max_{y\in\{0,1\}} f_i(y, \boldsymbol{x}), \quad i = 1, \ldots, m\,.$$

We refer to this method as RML, as it was originally called in (Petterson & Caetano, 2010).[1]

[1]Here, we follow the interpretation of this method as given by Petterson & Caetano (2011); in the original paper, it was introduced in a different setting.

An alternative approach was introduced by Petterson & Caetano (2011), in which $f(\boldsymbol{y}, \boldsymbol{x})$ additionally models assortative (submodular) pairwise interactions between labels:

$$f(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{m} f_i(y_i, \boldsymbol{x}) + \sum_{1\le y_k < y_l \le m} f_{k,l}(y_k, y_l)\,, \tag{13}$$

where the $f_{k,l}$ are non-negative and take into account the normalized count of co-occurrences of labels λ_k and λ_l. In this model, the prediction (9) can be accomplished exactly and efficiently via graph-cuts. However, the worst-case bound for graph-cut algorithms is $\mathcal{O}(m^3)$. The constraint generation problem (11) becomes more involved, too, and requires the solution of an intractable optimization problem. The authors propose an approximate algorithm working in $\mathcal{O}(m^4)$, and they prove the non-trivial guarantee that each label which is part of the solution is also part of the optimal solution. We refer to this method as SML, as originally called in (Petterson & Caetano, 2011).

7. Statistical Consistency

The algorithms introduced in previous sections do not directly minimize the F_β-loss. Instead, they either minimize a surrogate loss or follow a reduction scheme. The performance of these algorithms can be evaluated empirically for finite samples, using synthetic or real benchmark data. However, it is also interesting to analyze their infinite sample performance, by verifying whether they converge to the Bayes classifier for the F_β-measure. This type of consistency analysis has already been performed for different learning frameworks in general (Bartlett et al., 2006; Tewari & Bartlett, 2007) and for multi-label classification in particular (Gao & Zhou, 2011), but not yet for the F_β-measure and the algorithms discussed in this paper. More formally, we will use the following definition of *statistical consistency*.

Definition 1. *Given a surrogate loss $\tilde{\ell}(\boldsymbol{y}, f)$, a task loss $\ell(\boldsymbol{y}, \boldsymbol{h})$, and a prediction function $\boldsymbol{h} = \boldsymbol{h}(f)$, we say that $\tilde{\ell}$ is consistent with respect to ℓ, if for any distribution P and any sequence of classifiers $f_1, f_2, \ldots$ such that*

$$\lim_{n\to\infty} \tilde{L}(f_n, P) = \min_f \tilde{L}(f, P)\,,$$

we have

$$\lim_{n\to\infty} L(\boldsymbol{h}(f_n), P) = \min_{\boldsymbol{h}} L(\boldsymbol{h}, P)\,,$$

where $\tilde{L}(f, P) = \mathbb{E}[\tilde{\ell}(\boldsymbol{Y}, f(\boldsymbol{X}))]$.

There is an equivalent condition for consistency which is sometimes more convenient to use:

Theorem 1 (Gao & Zhou (2011)). *For any distribution P, let $A(P) = \arg\min_{\boldsymbol{h}} L(\boldsymbol{h}, P)$ be the set of all Bayes classifiers for loss function ℓ. A surrogate loss $\tilde{\ell}$ is consistent with respect to ℓ if and only if for all P:*

$$\min_f \tilde{L}(f,P) < \inf_f \left\{ \tilde{L}(f,P) \colon \boldsymbol{h}(f) \notin A(P) \right\} \qquad (14)$$

Although the above definition and theorem directly refer to the surrogate loss, we shall show later that they can also be applied to the plug-in rule approach. Starting with the analysis of the structured hinge loss, we find that both methods proposed in (Petterson & Caetano, 2010; 2011) are inconsistent, as stated formally in the theorem below. Since the feature vector $\boldsymbol{x}$ does not play any role in the following, we omit the dependence on $\boldsymbol{x}$ for the rest of the section.[2]

Theorem 2. *Consider the surrogate loss $\tilde{\ell}$ defined as in (10):*

$$\tilde{\ell}(\boldsymbol{y}, f) = \max_{\boldsymbol{y}' \in \mathcal{Y}} \{\ell_{F_\beta}(\boldsymbol{y}, \boldsymbol{y}') + f(\boldsymbol{y}')\} - f(\boldsymbol{y}) \,, \qquad (15)$$

for any $\boldsymbol{y} \in \mathcal{Y}$, and any real valued function $f \colon \mathcal{Y} \to \mathbb{R}$. Let $\boldsymbol{h}(f) = \arg\max_{\boldsymbol{y}} f(\boldsymbol{y})$ be a label vector predicted by f. Then $\tilde{\ell}$ is inconsistent with respect to ℓ_{F_β}. The inconsistency remains even if additional structure given by (12) or (13) is imposed on the function f.

Proof. For the sake of clarity, we show the theorem for $\beta = 1$, but a small modification of the proof will suffice to cover any $\beta > 0$. Let us choose $m = 2$, and the joint label distribution $p_{00} = 0.5$, $p_{01} = 0.2$, $p_{10} = 0$, $p_{11} = 0.3$, where $p_{uv} = P(y_1 = u, y_2 = v)$. Given a prediction $\boldsymbol{h}$ and a true label vector $\boldsymbol{y}$, the F_1-loss (3) is summarized by the following table:

	$\boldsymbol{h}=(0,0)$	$\boldsymbol{h}=(0,1)$	$\boldsymbol{h}=(1,0)$	$\boldsymbol{h}=(1,1)$
$\boldsymbol{y}=(0,0)$	0	1	1	1
$\boldsymbol{y}=(0,1)$	1	0	1	1/3
$\boldsymbol{y}=(1,0)$	1	1	0	1/3
$\boldsymbol{y}=(1,1)$	1	1/3	1/3	0

Thus, the Bayes classifier for the F_1-loss is $(0,0)$, whence $A(P) = \{(0,0)\}$. Now, for any f, the expected surrogate loss (15), $\tilde{L}(f,P)$, is given by

$$\begin{aligned} &p_{00}(\max\{0 + f_{00}, 1 + f_{01}, 1 + f_{10}, 1 + f_{11}\} - f_{00}) \\ &+p_{01}(\max\{1 + f_{00}, 0 + f_{01}, 1 + f_{10}, \tfrac{1}{3} + f_{11}\} - f_{01}) \\ &+p_{11}(\max\{1 + f_{00}, \tfrac{1}{3} + f_{01}, \tfrac{1}{3} + f_{10}, 0 + f_{11}\} - f_{11}) \,, \end{aligned}$$

where we denote $f_{uv} = f((u,v))$ and used the fact that $p_{10} = 0$. Now, we show that for any choice of f, the surrogate loss is at least 1. Since we can bound max from below by any of its terms, we can write

$$\begin{aligned} \tilde{L}(f,P) \geq\; & p_{00}(1 + f_{01} - f_{00}) + p_{01}(1 + f_{00} - f_{01}) \\ & + p_{11}(1 + f_{00} - f_{11}) = 1 + 0.3(f_{01} - f_{11}) \end{aligned}$$

On the other hand,

$$\begin{aligned} \tilde{L}(f,P) \geq\; & p_{00}(1 + f_{11} - f_{00}) + p_{01}(1 + f_{00} - f_{01}) \\ & + p_{11}(1 + f_{00} - f_{11}) = 1 - 0.2(f_{01} - f_{11}). \end{aligned}$$

The two inequalities above imply that $\tilde{L}(f,P) \geq 1$ for any f. Now, if we choose f, such that $f_{00} = -0.1$, $f_{01} = 0.1$, $f_{10} = -0.1$, $f_{11} = 0.1$, we have $\tilde{L}(f,P) = 1$, which means that f is a Bayes classifier for $\tilde{\ell}$. However, $\boldsymbol{h}(f) \in \{(0,1),(1,1)\}$, and thus $\boldsymbol{h}(f) \notin A(P)$, which violates (14). Moreover, it is easy to see that f as constructed above satisfies (12) and (13) if we set $f_1(y_1) = 0$, $f_2(y_2) = 0.1(2y_2 - 1)$, and $f_{k,l} \equiv 0$, so that inconsistency even holds under these constraints. □

Despite the fact that the plug-in rule approach defined in Section 5 may deliver conflicting estimates of $\boldsymbol{P}$ and $P(\boldsymbol{0} \,|\, \boldsymbol{x})$ for finite training data, we can show that it is consistent if the sample size grows to infinity.

Theorem 3. *Consider the surrogate loss $\tilde{\ell}(\boldsymbol{y}, f)$ obtained for the reduction to independent multinomial regression tasks:*

$$\tilde{\ell}(\boldsymbol{y}, \boldsymbol{f}) = \ell_{\log}([\![s_{\boldsymbol{y}} = 0]\!], f^0) + \sum_{i=1}^{m} \ell_{\log}(s_{\boldsymbol{y}}[\![y_i = 1]\!], f^i) \,,$$

where $\boldsymbol{f} = (f^0, \ldots, f^m)$ is a vector of real-valued functions $f^i \colon \{0, \ldots, m\} \to \mathbb{R}$. Let $\boldsymbol{h}(\boldsymbol{f})$ be the prediction from the GFM procedure, where all probabilities are obtained from $\boldsymbol{f}$ using the logistic transform as described in Section 5. Then $\tilde{\ell}$ is consistent with respect to ℓ_{F_β}.

Proof. Consider a standard multinomial logistic loss $\ell_{\log}(y, f)$ for multi-class classification, and let P_f be a probability estimate obtained from f through a logistic transform. It is easy to verify that $\mathbb{E}[\ell_{\log}(Y, f)] = H(P) + D(P \| P_f)$, where $H(P)$ is the entropy of the true distribution P, and $D(P \| P_f)$ is the relative entropy between P and P_f (Cover & Thomas, 1991). Thus, given any sequence $f_1, f_2, \ldots$ such that $\mathbb{E}[\ell_{\log}(Y, f_n)] \to \min_f \mathbb{E}[\ell_{\log}(Y, f)]$, we must have $D(P \| P_{f_n}) \to 0$, which, according to the Pinkser inequality, implies $P_{f_n} \to P$ in a total-variation sense.

We prove the theorem directly from Definition 1. Consider any sequence $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots$ such that $\tilde{L}(\boldsymbol{f}_n, P) \to \min_f \tilde{L}(\boldsymbol{f}, P)$. Since $\tilde{L}(\boldsymbol{f}_n, P)$ is a sum of expected logistic losses, convergence of $\tilde{L}(\boldsymbol{f}_n, P)$ to its minimum implies convergence of each expected logistic

[2] One may assume that everything is presented conditionally for a given $\boldsymbol{x}$.

loss to its respective minimum. This in turn implies that all probability estimates $P_{f_n^i}$ converge to the true probabilities P in a total-variation sense. Since each term under arg max in (6) is a continuous function of the probability estimates (and there in only a finite number of such terms), $\boldsymbol{h}(\boldsymbol{f}_n)$ will eventually be in $A(P)$ for sufficiently large n, i.e., the plug-in classifier will eventually agree with the Bayes classifier for the F_β-loss. This implies that $L_{F_\beta}(\boldsymbol{h}(\boldsymbol{f}_n), P) \to \min_{\boldsymbol{h}} L_{F_\beta}(\boldsymbol{h}, P)$. □

8. Experimental Results

We complement our theoretical analysis by an empirical evaluation of the methods on finite data sets. More specifically, we compare two plug-in rule methods, namely the Exact-F_β-Plug-in classifier (EFP) and the Label-independence-F_β-Plug-in classifier (LFP), and two structured loss minimization methods, namely RML and SML.[3] As an additional baseline, we include the so-called binary relevance (BR) approach that learns and predicts for each label independently. This algorithm essentially corresponds to LFP without the inference phase. We consider two performance measures, F_1 and Hamming loss, as well as running times of the training and inference procedures.

8.1. Setting

All approaches included in the comparison use base functions that are linear in the feature space. We train BR, EFP and LFP by using regularized multinomial regression.[4] We tune the regularization parameter λ for each base classifier independently by minimizing the logistic loss, which should provide better probability estimates. We use 5-fold cross-validation and choose λ from $\{10^{-4}, 10^{-3}, \ldots, 10^{3}\}$.

Similarly, RML has a single parameter λ, and we tune it in 5-fold cross-validation using the same range of values. SML has an additional parameter c that determines the trade-off between the linear f_i and the label interaction terms $f_{j,k}$ in (13). To guarantee a fair comparison, we only tune λ using 5-fold cross-validation for each experiment, whereas the best value of the parameter c was selected in an earlier series of experiments. Moreover, we use 5% of the label pairs for all datasets—according to Petterson & Caetano (2011), the results with other settings are very similar. The maximal number of iterations in the cutting-plane algorithm is set to 1000 for both RML and SML.

It needs to be mentioned that the plug-in rule approaches are implemented in Java, while the RML and SML are C++ programs. Therefore, the evaluation times may not be fully comparable. We run the experiments on a Debian virtual machine with 8-core x64 processor and 5GB RAM.[5] We use 6 benchmark datasets, which are publicly available from Mulan.[6] Table 1 provides a summary of some basic statistics of these datasets.

Table 1. Datasets and their properties: the number of training (#TRAIN) and test (#TEST) examples, the number of labels (m) and features (d), and the average number of classes (q) in a single multinomial regression task of EFP.

DATASET	#TRAIN	#TEST	m	d	q
IMAGE	1200	800	5	135	4
SCENE	1211	1196	6	294	3.33
YEAST	1500	917	14	103	10.36
MEDICAL	333	645	45	1449	2.466
ENRON	1123	579	53	1001	6.58
MEDIAMILL	30993	12914	101	120	11.41

8.2. Results

The results are summarized in Table 2. As can be seen, all methods tailored for the F_1-measure outperform the baseline on this measure, whereas BR achieves the best results for Hamming loss. This is coherent with the result of Gao & Zhou (2011), according to which this approach is consistent for this loss function. With respect to the F_1-measure, the best method is EFP that wins on five out of six datasets. To some extent, this result can be explained by our theoretical results regarding the consistency of the methods.

In general, it seems that the plug-in rule methods are superior to the methods based on structured loss minimization. Among the latter, RML outperforms SML.[7] This is in agreement with the original results of Petterson & Caetano (2011), suggesting that SML achieves better results only for datasets with a reduced number of features. One may conjecture that the label interaction terms are more helpful in properly modeling the feature space than in optimizing the F_β-measure.

[3] For both RML and SML, we use the implementation offered by the authors (Petterson & Caetano, 2011) available at `http://users.cecs.anu.edu.au/~jpetterson/`.

[4] We use the implementation of multinomial regression by Mallet (McCallum, 2002) available at `http://mallet.cs.umass.edu/`.

[5] Remark that all methods can be easily parallelized. All subtasks of LFP and EFP can be trained independently. The implementation of RML and SML is also multi-threaded (Petterson & Caetano, 2011). We used, however, the sequential variants of the algorithms to simplify the comparison.

[6] `http://mulan.sourceforge.net/datasets.html`

[7] The results we obtained for RML are at least as good as the ones presented in (Petterson & Caetano, 2011), while for SML are slightly worse.

Table 2. Experimental results for Hamming loss (HL), F_1, and running times (in seconds) of cross-validation (t_{cv}), training (t_{train}) (for the best set of parameters) and inference (t_{inf}). The best results are marked by a '*'.

	HL[%]	F[%]	t_{cv}	t_{train}	t_{inf}	HL[%]	F[%]	t_{cv}	t_{train}	t_{inf}	HL[%]	F[%]	t_{cv}	t_{train}	t_{inf}
	IMAGE					SCENE					YEAST				
BR	*19.90	43.63	*9	*0.392	0.087	10.51	55.73	*29	*0.733	0.241	*20.03	60.59	*26	*0.901	0.128
LFP	27.55	58.86	*9	*0.392	0.119	12.18	74.38	*29	*0.733	0.270	22.24	65.02	*26	*0.901	0.146
EFP	26.07	*59.77	24	0.606	0.183	12.22	*74.44	72	0.995	0.399	22.82	*65.47	101	2.004	0.367
RML	25.07	57.49	94	1.104	*0.051	*9.70	73.92	73	1.001	*0.118	22.82	64.78	206	5.194	*0.056
SML	28.82	56.99	156	7.116	0.052	15.65	68.50	52	1.129	0.123	24.52	63.96	319	4.385	0.070
	MEDICAL					ENRON					MEDIAMILL				
BR	*1.17	70.19	*9	*1	0.952	*4.54	55.49	*52	*4	1.016	*3.19	51.21	*3238	*118	13
LFP	1.18	*81.27	*9	*1	1.513	6.09	56.86	*52	*4	1.519	3.67	55.15	*3238	*118	20
EFP	1.23	80.39	16	1	1.883	5.34	*61.04	214	6	2.628	3.63	*55.16	24620	440	30
RML	1.20	80.63	1253	30	*0.144	6.35	57.69	3897	41	*0.143	4.12	49.35	–	1125	*7
SML	2.50	67.90	715	23	0.773	7.82	54.61	18780	62	0.887	4.18	50.02	–	10365	131

Table 2 also shows the runtimes for parameter tuning in cross-validation (t_{cv}), training for the best set of parameters, and inference. As we mentioned above, the comparison of the running times should be interpreted with caution, due to the use of different programming languages and differences in the implementations. For example, the inference times for BR and RML should basically be very similar, as in both cases we apply m linear models. Yet, the implementation of RML is much more efficient. Nevertheless, we are still able to derive several important conclusions.

RML is most efficient in inference, which is coherent with our analysis. Nevertheless, the inference times of the plug-in rule approaches are quite comparable to those of BR, despite their quadratic (for LFP) and cubic (for EFP) complexity. Admittedly, however, the datasets used in the experiments only contain a small to moderate number of labels (up to 100). For datasets with thousands of labels, the difference is likely to become substantially larger. The inference for SML is slower than for RML, but for all datasets except one, it is faster than the plug-in rule classifiers. For the MEDIAMILL dataset, this method takes the longest time.

The training of BR and LFP (these are exactly the same procedures) is the most effective. Training of EFP leads to m multinomial regression models. One should note, however, that the average number of classes (column q in Table 1) for all datasets is much smaller than the highest possible value $m + 1$. Therefore, the training of EFP is still quite effective and takes only a few times longer than the training of LFP. The cutting-plane algorithm and the constraint generation step slow down the training of RML and SML, and SML performs worst in this regard. Still, the cutting-plane algorithm converges very fast in several cases.

The plug-in rule approaches may tune the parameters internally for each subtask independently, without explicitly running the inference step. Therefore, the tuning times of LFP and EFP are also better than those of RML and SML. For the MEDIAMILL dataset, RML and SML were not able to perform the parameter tuning step in a reasonable amount of time. To get the results for this dataset, we trained these methods with different settings and selected the best result on the test set. We also reduced the maximum number of iterations to 200.

9. Conclusion

We discussed and analyzed two conceptually different approaches to F_β-measure maximization (F_β-loss minimization) in multi-label classification. The plug-in rule methods estimate all parameters that are needed to compute the prediction of the Bayes classifier, whereas methods based on structured loss minimization, such as structured SVMs, produce a classifier more directly through minimization of the loss on the training data.

Moreover, we introduced a novel plug-in rule algorithm that performs parameter estimation via a set of multinomial regression tasks. Theoretically, we have shown this algorithm to be consistent for the F_β-measure, whereas the SSVM approach is not consistent. This result is corroborated by our experimental studies, in which the plug-in rule approach performs particularly well.

Acknowledgments. The first three authors are supported by the Foundation of Polish Science under the Homing Plus programme, co-financed by the European Regional Development Fund. The last author is supported by German Research Foundation. We thank all anonymous reviewers for their valuable comments.

References

Bartlett, P., Jordan, M., and Mcauliffe, J. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

Chai, K. Expectation of F-measures: Tractable exact computation and some empirical observations of its properties. In *SIGIR*, 2005.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. John Wiley, 1991.

Dembczyński, K., Waegeman, W., Cheng, W., and Hüllermeier, E. An exact algorithm for F-measure maximization. In *Advances in Neural Information Processing Systems*, volume 25, 2011.

Gao, W. and Zhou, Z. On the consistency of multi-label learning. In *COLT*, 2011.

Hariharan, B., Vishwanathan, S. V. N., and Varma, M. Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine Learning Journal*, 88(1):127–155, 2012.

Jansche, M. A maximum expected utility framework for binary sequence labeling. In *ACL 2007*, pp. 736–743, 2007.

Kelley, J. E. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, 8:704–712, 1960.

Lewis, D. Evaluating and optimizing autonomous text classification systems. In *SIGIR 1995*, pp. 246–254, 1995.

McCallum, A. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

Petterson, J. and Caetano, T. S. Reverse multi-label learning. In *Advances in Neural Information Processing Systems 24*, pp. 1912–1920, 2010.

Petterson, J. and Caetano, T. S. Submodular multi-label learning. In *Advances in Neural Information Processing Systems 24*, pp. 1512–1520, 2011.

Quevedo, J., Luaces, O., and Bahamonde, A. Multilabel classifiers with a probabilistic thresholding strategy. *Pattern Recognition*, 45, 2012.

Tewari, A. and Bartlett, P. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, May 2007.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6: 1453–1484, 2005.

Ye, N., Chai, K., Lee, W., and Chieu, H. Optimizing F-measures: a tale of two approaches. In *ICML*, 2012.