Mach Learn (2015) 98:435–454
DOI 10.1007/s10994-014-5439-y

Efficient *F* measure maximization via weighted maximum likelihood

Georgi Dimitroff · Georgi Georgiev · Laura Toloşi · Borislav Popov

Received: 5 September 2013 / Accepted: 13 March 2014 / Published online: 10 April 2014

Abstract The classification models obtained via maximum likelihood-based training do not necessarily reach the optimal F_β-measure for some user's choice of β that is achievable with the chosen parametrization. In this work we link the weighted maximum entropy and the optimization of the expected F_β-measure, by viewing them in the framework of a general common multi-criteria optimization problem. As a result, each solution of the expected F_β-measure maximization can be realized as a weighted maximum likelihood solution within the maximum entropy model - a well understood and behaved problem for which standard (off the shelf) gradient methods can be used. Based on this insight, we present an efficient algorithm for optimization of the expected F_β using weighted maximum likelihood with dynamically adaptive weights.

1 Introduction and related work

The F measure is a common tool for expressing a trade-off between precision and recall in binary classification, which might be specific to each application. Optimizing an F measure based performance of a classifier has a long history and many applications in text analysis. High precision is often needed in the analysis of clinical records, job boards, automatic summarization, machine translation etc., whereas high recall is usually preferred when the output of the system will be used in information retrieval applications, as an input to other text

Editor: Shai Shalev-Shwartz.

G. Dimitroff (✉) · G. Georgiev · L. Toloşi · B. Popov
Ontotext AD, Sofia, Bulgaria
e-mail: georgi.dimitrov@ontotext.com

G. Georgiev
e-mail: georgi.georgiev@ontotext.com

L. Toloşi
e-mail: laura.tolosi@ontotext.com

B. Popov
e-mail: borislav.popov@ontotext.com

analysis components like relation extraction (Georgiev et al. 2009), correference (Ganchev et al. 2007; Carpenter 2007), or software-aided curation and annotation (Ganchev et al. 2007).

Typically the Precision/Recall preferences are expressed by requiring a large F_β measure, the weighted harmonic mean of precision and recall, for some specific β, where β encodes the particular Precision/Recall trade-off. Obviously, the larger the β the more preference is put on optimizing the precision as compared to the recall. In order to avoid confusions with other equivalent definitions, we specify that in this article we define F_β as

$$F_\beta = (\beta/\text{Precision} + (1-\beta)/\text{Recall})^{-1}, \text{ for some} \beta \in [0, 1].$$

In some learning algorithms the precision/recall trade-off have also been controlled by means of different loss functions. For instance, Ganchev et al. (2007) use the MIRA (Crammer et al. 2006) learning algorithm to boost the recall of a gene mention tagger, by using a loss that is a weighted combination of the number of false positive and false negative gene mentions in a sentence. Other authors focus on both precision and recall and develop methods for multiobjective optimization of both scores. The approach is preceded by likelihood training (Klinger 2009).

Maximum-likelihood-based binary classifiers such as maximum entropy are relatively easy to fit, but they are rigid and cannot be tuned to a desired Precision and Recall trade-off. In this work, we show that the well known weighted MaxEnt (Vandev and Neykov 1998), with corresponding estimation that is barely harder than in the standard case without weights, can be used to optimize the F_β measure; see also Simecková (2005) for an interesting discussion on weighted maximum likelihood. The main statement of the paper is that if appropriate weights are chosen, then the fitted maximum weighted likelihood model coincides with the optimal expected F_β model for the binary classification task. There are at least two major advantages of the weighted likelihood as a loss function, namely: (i) it is concave and (ii) standard (off the shelf) gradient methods can be used for its optimization. To the best of our knowledge, such a link between the weighted maximum likelihood and the F_β maximization has not been established before. The article is focused on the intuition of the relation and the proof of the main result. The value of our theoretical observation is that it establishes the methodology of viewing a particular model as a specific solution of a common multi-criteria optimization problem. Furthermore, we give a handy expression for the optimal weights and propose and test an efficient algorithm for the maximization of the expected F_β measure. Obviously if the assumed model is, in contrast to the considered in this article maximum entropy framework, such that the (weighted) maximum likelihood estimation is computationally not accessible there is little gain in casting the hard problem of F_β maximization into a possibly even harder maximum likelihood problem.

This article is organized as follows: in Sect. 2 we briefly introduce the weighted maximum entropy model. In Sect.3, the F_β measure as a trade-off of precision and recall is presented and the expected F_β is defined (see also Nan et al. 2012). In Sect. 4, we establish the link between the optimal expected F_β and a weighted maximum entropy model. Sections 5 and 6 present a method for estimating weights that afford optimal expected F_β and a corresponding algorithm. We describe our approach for evaluation of the algorithm in Sect. 7. In Sect. 8 we introduce the datasets that we used for experiments. Section 9 presents the results and Sects. 10 and 11 conclude the article with comments and a discussion.

Related work One of the most popular heuristics for precision-recall trade-off is based on adjusting the acceptance threshold given by maximum entropy models (or any other model that estimates class posterior probabilities). However, this procedure amounts to a simple translation of the maximum likelihood hyperplane towards or away from the target class and does not fit the model anew. Thresholding on posterior probabilities has been used in the context of other learning frameworks, such as Conditional Random Fields (CRFs) (Culotta 2004).

Minkov et al. (2006) introduces another heuristic, which is based on changing the weight of a special feature, which indicates if a sample is in the sought-after class or not.

Jansche (2005) describes a maximum entropy model that optimizes directly an expected F_β-based loss by means of gradient ascent. However the expected F_β is not concave and is rather cumbersome to deal with. Therefore the standard gradient methods do not guarantee optimality of the solution.

Approaches for training CRFs (Lafferty 2001) to directly optimize multivariate evaluation measures, including non-linear measures such as the F-score (Suzuki et al. 2006) have been proposed recently.

Joachims (2005) proposes a multivariate SVM capable of optimizing an upper bound of a class of rather general nonlinear performance measures including F_β. The obtained optimization problem is of exponential complexity, but the solution can be approximated in polynomial time using a heuristic algorithm.

Our approach is somewhat different in nature - instead of directly optimizing the nonlinear performance measure, we show that the optimizer can be represented as a maximizer of a weighted version of a traditional "linear" performance measure, i.e. one that decomposes into a sum of performance measures over each training example. Needless to say, we also keep the parametrization of the conditional probabilities and in fact optimize the expected F_β measure rather than an upper bound.

A general algorithm for F-measure optimization is given in Dembczyn'ski et al. (2011), however they rely on known data distributions, which is an unrealistic requirement in practice. A very interesting result in Dembczyn'ski et al. (2011) is that there is a lower bound on the discrepancy between the optimal solution and the solution obtained by means of optimal acceptance threshold.

In Nan et al. (2012), the authors systematically classify the existing approaches for F_β maximization into two groups: empirical utility maximization (EUM) and decision-theoretic approaches (DTA). The authors show that EUM-optimal and DTA-optimal algorithms are asymptotically equivalent and also comment on the practical advantages and disadvantages of the two classes of approaches. In their article when considering the EUM approaches, where in general our framework resides, they basically restrict to approaches where a score function (e.g. parametric model for the conditional probabilities) is learned and only the acceptance threshold is obtained by directly optimizing the F-measure on the expected F measure $\tilde{F}$ is also considered in Nan et al. (2012), where also its consistency is stated and even a Hoeffding bound for the convergence is given.

2 The maximum entropy model

The maximum entropy modeling framework as introduced in the NLP domain in Berger et al. (1996) has become standard for various NLP tasks. To fix notations consider a training set of m examples $\{(x_i, y_i) : i \in 1, \ldots m\}$ where x_i's are the attributes and the y_i's are the classes taking values in some finite set $\mathcal{Y}$. Each observation is represented by a set of N features $\{f_j(x_i, y_i) : j \in 1, \ldots, N\}$.

The maximum entropy principle forces the model conditional probabilities $p(y \mid x, \lambda)$ of an example with attributes x to be of class y to be of the form

$$p(y \mid x, \lambda) = \frac{1}{Z_\lambda(x)} \exp(\lambda \cdot f),$$

where $\lambda \in \mathbb{R}^N$ are the parameters of the model and $Z_\lambda(x)$ is the corresponding partition function given by

$$Z_\lambda(x) = \sum_{y \in \mathcal{Y}} \exp(\lambda \cdot f).$$

The calibration of the model amounts to (see Berger et al. 1996) maximizing the log-likelihood function

$$l(\lambda : x, y) = \sum_{i=1}^{m} \log p(y_i \mid x_i, \lambda). \tag{1}$$

In the following for a weight vector $w \in \mathbb{R}^m$ we will make use of the weighted log-likelihood function

$$l^W(\lambda : w, x, y) = \sum_{i=1}^{m} w(i) \log p(y_i \mid x_i, \lambda). \tag{2}$$

Apart from the obvious technical generalization of the likelihood function the weights could also be interpreted as modifying the training set by adding new examples having the same attributes and classes (x_i, y_i) with intensity $w(i)$ resulting in an expected number of $w(i)$ identical training examples. In particular for $w(i) < 1$ the i-th example is deleted with probability $1 - w(i)$. If $w(i) > 1$, say $w(i) = z + w_f(i)$ for some integer $z \geq 1$ and $0 \leq w_f(i) < 1$ then generate z training identical examples (x_i, y_i) and additionally clone the ith example with probability $w_f(i)$.

In what follows we will always assume that the underlying log-likelihood function has a maximum. A possible exception is when the training set is separable in which case the log-likelihood converges to zero while the parameters diverge. The same holds for the weighted log-likelihood as long as the weights are strictly positive which will be the case in this paper. To see that indeed the weighted log-likelihood has a maximum when the training set is not separable we refer again to the interpretation of the weights as a modification of the training set. Indeed if the weights are integers then the weighted log-likelihood is in fact the standard log-likelihood for the modified training set where each training example (x_i, y_i) is duplicated w_i times. Hence the weighted log-likelihood attains its maximum if the modified training set is not separable which is exactly then the case when the original set is not separable. The case of rational weights can be reduced back to the integer case by a simple scaling of the log-likelihood with an appropriate integer, and the general case with real weights can be tackled via approximating it with weighted log-likelihoods having rational weights. Hence, as long as the weights are strictly positive, assuming the non-separability of the training set yields the existence of the unique maximum of the weighted log-likelihood. The case when the training set is separable is not relevant for our considerations here because in that case any separating plane results in an maximal F_β measure for any β.

In this paper we are aiming at optimizing the F measure which is only defined in the binary classification case. Therefore, for the rest of the paper we restrict discussion to the binary classification case with $|\mathcal{Y}| = 2$, i.e. the case where we have only two classes, which we will denote with u and $\bar{u}$. Hence, from now on $y_i \in \mathcal{Y} = \{u, \bar{u}\}$.

The extension to the multi-class case is possible but nontrivial and will be presented in a separate paper.

3 The precision/recall trade off expected F-measure

The learned maximum entropy model produces for each vector of attributes x a whole probability distribution $p(y \mid x, \lambda)$ over the space of classes $\mathcal{Y}$ giving the model conditional probabilities of the example being in each one of the classes. However when used for classification one typically would use the model as a maximum a-posteriori classifier, that is one would classify x in the class $y(x)$ that maximizes the model conditional probability:

$$y(x) = \text{argmax}_y p(y \mid x, \lambda).$$

The performance of the classifier for the class u is typically measured in terms of the *Precision* and *Recall* for the class u ($\bar{u}$ denotes the complementary class) defined as

$$\begin{aligned} P &= \frac{\text{\#true u}}{\text{\#true } u + \text{\#false } u} =: \frac{A_u}{A_u + C_u} \\ R &= \frac{\text{\#true } u}{\text{\#true } u + \text{\#false } \bar{u}} =: \frac{A_u}{A_u + B_u} \end{aligned} \tag{3}$$

i.e. the precision is the ratio between the number of true positives divided by all examples classified as positive, while the Recall is the the ratio of the number of true positives divided by the number of all positive examples. For a class u, called positive, we will denote with A_u, B_u, C_u and D_u the number of true u, false negative, false u and true negative classifications respectively.

Typically if the model is not overfitting we cannot maximize simultaneously both the precision and recall and dependent on the application one typically focuses on a particular trade-off between them described by a constant $\beta \in [0, 1]$ and expressed as the β-weighted harmonic mean called F_β-measure:

$$F_\beta := \left(\frac{\beta}{P} + \frac{1-\beta}{R}\right)^{-1}.$$

The larger the β the greater the influence of the Precision as compared to the Recall on the F_β-measure.

Expected F-measure: The problem with the maximum posterior probability classification is that it only takes into account the index of the largest probability in the vector of model probabilities and completely disregards the rest. Instead we focus on the stochastic classifier which takes into account the whole information contained in the learned model. More precisely, an example with a given vector of attributes x is classified into the class $y(x)$ which is randomly drawn from the set of all classes according to the model conditional probabilities $p(y \mid x, \lambda)$. In this set-up the quantities A_u, B_u, C_u and D_u become random variables. However if we repeatedly perform the stochastic classification and average over the results we would get approximately their expected values over the training set which we denote with $\tilde{A}_u$, $\tilde{B}_u$, $\tilde{C}_u$, $\tilde{D}_u$ respectively:

$$\begin{aligned}
\tilde{A}_u &= \mathbb{E}\#\text{true } u = \sum_{i:y_i=u} p(u \,|\, x_i, \lambda); \\
\tilde{B}_u &= \mathbb{E}\#\text{false } \bar{u} = \sum_{i:y_i=u} \underbrace{(1 - p(u \,|\, x_i, \lambda))}_{=:p(\bar{u} \,|\, x_i, \lambda)}; \\
\tilde{C}_u &= \mathbb{E}\#\text{false } u = \sum_{i:y_i=\bar{u}} p(u \,|\, x_i, \lambda); \\
\tilde{D}_u &= \mathbb{E}\#\text{true } \bar{u} = \sum_{i:y_i=\bar{u}} (1 - p(u \,|\, x_i, \lambda));
\end{aligned} \tag{4}$$

With these expected counts we can define the plug-in estimators $\tilde{P}$ and $\tilde{R}$ of the precision and recall by simply interchanging the counts with their expected values. The corresponding approximation $\tilde{F}_\beta$ of the F_β measure is then defined using $\tilde{P}$ and $\tilde{R}$:

$$\tilde{F}_\beta = \left(\frac{\beta}{\tilde{P}} + \frac{1-\beta}{\tilde{R}} \right)^{-1}. \tag{5}$$

As in Jansche (2005), with a slight abuse of notation we will call the obtained plug-in approximation $\tilde{F}_\beta$ the expected F_β measure.

For a large training set and a good model, the expected F_β measure on the training set will be close to the standard one since most of the model probabilities $p(y_i \,|\, x_i, \lambda)$ will be close to 1 for the training examples.

In the next section we will show that we can maximize the expected F_β measure by modifying the training set via adding weights and then again estimating the model parameters using the well-behaved and understood maximum likelihood procedure.

4 Achieving expected F-measure maximization via weighted maximum likelihood

Clearly, the log-likelihood (1) and the expected F_β (5) are different, though - one would hope - not orthogonal objectives.

Intuitively, every reasonable machine learning model would try to set the model parameters λ in such a manner that for all training examples the model conditional probabilities of the observed classes y_i given the example's attributes x_i, namely $p(y_i \,|\, x_i, \lambda)$, are as large as possible. In general, if the used model is not overfitting badly it will not be possible for all conditional probabilities to be close to 1 simultaneously. Every particular model can be viewed as a specific method to implicitly handle these trade-offs. In this view the crucial difference between the log-likelihood and the expected F_β measure seen as objective functions is that while the log-likelihood approach gives equal importance to all training examples on the logarithmic scale the (expected) F_β measure has a parameter β controlling this trade-off on a class-wise scale. On the other hand as noted in Jansche (2005) the flexibility in $\tilde{F}_\beta$ comes at a price - the $\tilde{F}_\beta$ is by far not that nice function to optimize as the log-likelihood is. Fortunately, the $\tilde{F}_\beta$ maximization can be viewed in terms of the weighted likelihood maximization, which is only a slight generalization of the standard maximum likelihood. To make the above discussion precise we will rely on some basic facts from the theory of multi-criteria optimization. For a thorough and up to date treatise on the topic see Ehrgott (2005).

We will briefly give a definition of Pareto optimality (also sometimes called efficiency). For a multicriteria optimization problem (MOP)

$$max\{f_1(x), \dots, f_k(x)\} \text{ subject to } x \in \chi$$

the point $x_0 \in \chi$ is said to dominate $x_1 \in \chi$ if for all $i \in 1, \dots, k$ it holds that $f_i(x_0) \geq f_i(x_1)$ and for at least one $j \in 1, \dots, k$ $f_j(x_0) > f_j(x_1)$. A feasible point $x_0 \in \chi$ is called Pareto optimal if there is no point $x \in \chi$, which dominates x_0. The Pareto optimal set is the set of all Pareto optimal points. In plain language, Pareto optimality for a point x_0 means that by moving away from x_0 we cannot improve all objectives but rather we would necessarily deteriorate at least one objective.

The following proposition and in particular its proof make the above discussion precise by viewing the weighted maximum likelihood solution and the maximizer of the expected F_β measure as particular elements of the Pareto optimal set of the single multi-criteria optimization problem (MOP)

$$\max_\lambda \{p(y_1 \mid x_1, \lambda), \dots, p(y_m \mid x_m, \lambda)\}.$$

This way we unify the two objectives by viewing them as methods to pick a particular point from the Pareto optimal set associated with the common MOP stated above. Clearly, the approach is much more general and can be applied to other statistical machine learning models and objective functions as well.

As a simple consequence we obtain that each expected F_β measure maximizer can be realized as a weighted maximum likelihood estimator and even approximated via a class-wise weighted maximum likelihood estimator. Later on, we elaborate on the optimal weights.

Proposition 1 *Let $\hat{\lambda}_\beta$ be the maximizer of the expected F_β measure with respect to the class $u \in \mathcal{Y}$. Then there exists a vector of weights $w(\beta) \in \mathbb{R}^m$ such that $\hat{\lambda}_\beta$ coincides with the weighted maximum likelihood estimator*

$$\hat{\lambda}_{ML}^{w(\beta)} = arg \max l^W(\lambda : w(\beta), x, y).$$

That is we have

$$\hat{\lambda}_\beta = \hat{\lambda}_{ML}^{w(\beta)}.$$

We first sketch the *idea of the proof:*

The maximum likelihood optimizes simultaneously the conditional probabilities

$p(y_i \mid x_i, \lambda)$ via implicitly setting some trade-offs between them. Therefore our idea is to adjust these trade-offs using the weights in such a manner that the F_β is optimized rather than the rigid likelihood. The most natural and general way to look at these trade offs is to consider the MOP (multicriteria optimization problem)

$$\max\{\log p(y_1 \mid x_1, \lambda), ..., \log p(y_m \mid x_m, \lambda)\}.$$

It turns out that both the max likelihood and the expected F_β optimizer are particular solutions of the MOP above. On the other hand all solutions of the MOP can be obtained by maximizing nonnegative linear combinations of the objectives (see Theorem 3.15 in Ehrgott 2005). However a nonnegative combination of the objectives $\log p(y_i \mid x_i, \lambda)$ is precisely a weighted maximum entropy objective function.

Proof Let $\hat{\lambda}_\beta$ be the $\tilde{F}_\beta$ maximizer, i.e.

$$\hat{\lambda}_\beta = \text{argmax}_\lambda \tilde{F}_\beta.$$

We first rewrite the expected F_β measure as follows:

$$\tilde{F}_\beta = \left(\frac{\beta}{\tilde{P}} + \frac{1-\beta}{\tilde{R}}\right)^{-1} = \frac{\tilde{A}_u}{\beta(\tilde{A}_u - \tilde{D}_u) + (1-\beta)m_u + \beta m_{\bar{u}}}, \tag{6}$$

where m_u and $m_{\bar{u}}$ denote the total number of training examples in classes u and $\bar{u}$ respectively.

Equation (6) (observe the denominator is always positive) shows that the maximizer $\hat{\lambda}_\beta$ of $\tilde{F}_\beta$ is an element of the Pareto optimal set of the MOP

$$\max_\lambda \{\tilde{A}_u, \tilde{D}_u\}. \tag{7}$$

Indeed, if we would assume that $\hat{\lambda}_\beta$ is not Pareto efficient for the MOP (7) above then we can find another set of parameters λ_0 such that the pair $(\tilde{A}_u(\lambda_0), \tilde{D}_u(\lambda_0))$ dominates $(\tilde{A}_u(\hat{\lambda}_\beta), D_u(\hat{\lambda}_\beta))$, that is at least one of the objectives is improved in λ_0 as compared with $\hat{\lambda}_\beta$ and the other one is at least not deteriorating. Since the function

$$f(x, y) = \frac{x}{\beta(x-y) + (1-\beta)m_u + \beta m_{\bar{u}}}$$

is increasing in x and y as long as the denominator is positive this would mean that $\tilde{F}_\beta(\lambda_0) > \tilde{F}_\beta(\hat{\lambda}_\beta)$ which contradicts the assumption that $\hat{\lambda}_\beta$ maximizes the expected F_β measure.

With a similar argument we can pass to the finer granularity MOP by observing that the Pareto optimal set of (7) is contained in the Pareto optimal set of the finer granularity MOP

$$\max_\lambda \{p(y_1 \,|\, x_1, \lambda), \ldots, p(y_m \,|\, x_m, \lambda)\}. \tag{8}$$

This follows from the fact that

$$\tilde{A}_u(\lambda) = \sum_{i:y_i=u} p(u \,|\, x_i, \lambda) \text{ and } \tilde{D}_u(\lambda) = \sum_{i:y_i=\bar{u}} p(\bar{u} \,|\, x_i, \lambda).$$

Indeed if we assume that λ is Pareto optimal for (7) but not for (8) then we can find λ_0 such that some of the objectives of (8) are improved and none of them is decreased. But this would mean that the pair $(A_u(\lambda_0), D_u(\lambda_0))$ dominates $(A_u(\lambda), D_u(\lambda))$, that is at least one of the strict inequalities $A_u(\lambda_0) > A_u(\lambda)$ and $D_u(\lambda_0) > D_u(\lambda)$ holds and at the same time both inequalities $A_u(\lambda_0) \geq A_u(\lambda)$ and $D_u(\lambda_0) \geq D_u(\lambda)$ hold, which is a contradiction with the assumption that λ is Pareto optimal for (7).

By now we know that the expected F_β optimizer $\hat{\lambda}_\beta$ is Pareto optimal for (8). Furthermore the Pareto efficient sets of (8) and the yet another MOP

$$\max_\lambda \{\log p(y_1 \,|\, x_1, \lambda), \ldots, \log p(y_m \,|\, x_m, \lambda)\} \tag{9}$$

coincide as the logarithm is strictly increasing function. Therefore $\hat{\lambda}_\beta$ is also Pareto optimal for (9). But since all of the objectives $\log p(y_i \,|\, x_i, \lambda)$ are concave in λ each of the Pareto efficient points of (9) can be realized as a maximizer of some nonnegative linear combination of the objectives $\log p(y_i \,|\, x_i, \lambda)$. For a proof of this result see Geoffrion (1968) or Theorem 3.15 in Ehrgott (2005) (there they have minimizers and respectively convex objective functions while we have maximizers and concave objectives). In particular, there are positive weights $(w(\beta)_1, \ldots, w(\beta)_m)$ such that

$$\hat{\lambda}_{F_\beta} = \text{argmax}_\lambda \left[\sum_{i=1}^m w(\beta)_i \log p(y_i \,|\, x_i, \lambda)\right] = \text{argmax}_\lambda l^W(\lambda : w(\beta), x, y).$$

$l^W(\lambda : w(\beta), x, y)$ is the weighted log-likelihood function and its maximizer is

$$\hat{\lambda}^w_{ML} := \text{argmax}_\lambda l^W(\lambda : w(\beta), x, y).$$

This concludes the proof. □

5 The weights

Following the lines of the proof, for a given objective β we can in fact calculate the weights $w(\beta)$ that would allow us to compute the expected F_β maximizer as a $w(\beta)$-weighted maximum likelihood. We also present a class-wise approximation $\bar{w}(\beta)$ for the instant-wise weights $w(\beta)$

In the proof of Proposition 1 we noticed that $\hat{\lambda}_\beta$, the maximizer of $\tilde{F}_\beta$, can as well be realized as

$$\hat{\lambda}_\beta = \text{argmax}_\lambda \left(\sum_{i=1}^m w(\beta)_i \log p(y_i \,|\, x_i, \lambda) \right) =: \text{argmax}_\lambda G(\lambda). \tag{10}$$

This means that both objective functions $\tilde{F}$ and the weighted log-likelihood G should have vanishing gradients at the optimal $\lambda = \hat{\lambda}_\beta$. For the gradients we have:

$$\nabla_\lambda \tilde{F}_\beta(\hat{\lambda}_\beta) = \sum_{i:y_i=u} \partial_{\tilde{A}} \tilde{F}(\hat{\lambda}_\beta)\, \nabla_\lambda p(y_i \,|\, x_i, \hat{\lambda}_\beta) + \sum_{i:y_i=\bar{u}} \partial_{\tilde{D}} \tilde{F}(\hat{\lambda}_\beta)\, \nabla_\lambda p(y_i \,|\, x_i, \hat{\lambda}_\beta)$$

$$\nabla_\lambda G(\hat{\lambda}_\beta) = \sum_{i=1}^m \frac{w(\beta)_i}{p(y_i \,|\, x_i, \hat{\lambda}_\beta)} \nabla_\lambda p(y_i \,|\, x_i, \hat{\lambda}_\beta).$$

If we choose some $c > 0$ and weights $w(\beta)$ such that for every i we have

$$c\, \frac{w(\beta)_i}{p(y_i \,|\, x_i, \hat{\lambda}_\beta)} = \begin{cases} \partial_{\tilde{A}} \tilde{F}(\hat{\lambda}_\beta) & \text{if } y_i = u \\ \partial_{\tilde{D}} \tilde{F}(\hat{\lambda}_\beta) & \text{if } y_i = \bar{u} \end{cases},$$

then we shall obtain

$$\nabla_\lambda G(\hat{\lambda}_\beta) = \sum_{i=1}^m \frac{w(\beta)_i}{p(y_i \,|\, x_i, \hat{\lambda}_\beta)} \nabla_\lambda p(y_i \,|\, x_i, \hat{\lambda}_\beta) = \frac{1}{c} \nabla_\lambda \tilde{F}_\beta(\hat{\lambda}_\beta) = 0.$$

Thus by setting

$$w(\beta)_i = \begin{cases} \frac{1}{c} p(u \,|\, x_i, \hat{\lambda}_\beta)\, \partial_{\tilde{A}} \tilde{F}(\hat{\lambda}_\beta) & \text{if } y_i = u \\ \frac{1}{c} p(\bar{u} \,|\, x_i, \hat{\lambda}_\beta)\, \partial_{\tilde{D}} \tilde{F}(\hat{\lambda}_\beta) & \text{if } y_i = \bar{u} \end{cases}$$

we obtain a weighted maximum-likelihood objective whose unique maximum is at $\hat{\lambda}_\beta$.

For the sake of simplicity we set $c = \partial_{\tilde{D}} \tilde{F}(\hat{\lambda}_\beta)$ (recall that $\tilde{F}_\beta$ is increasing in $\tilde{D}$ and hence $\partial_{\tilde{D}} \tilde{F}(\hat{\lambda}_\beta) > 0$) and the weights become

$$w(\beta)_i = \begin{cases} p(u \,|\, x_i, \hat{\lambda}_\beta)\, \frac{\partial_{\tilde{A}} \tilde{F}}{\partial_{\tilde{D}} \tilde{F}}(\hat{\lambda}_\beta) & \text{if } y_i = u \\ p(\bar{u} \,|\, x_i, \hat{\lambda}_\beta) & \text{if } y_i = \bar{u} \end{cases}.$$

Using expression (6) for the expected F_β and some simple algebra we get

$$\frac{\partial_A \tilde{F}_\beta}{\partial_D \tilde{F}_\beta} = \frac{\tilde{z}_u - \beta \tilde{D}_u}{\beta \tilde{A}_u} = \frac{1 - \beta \tilde{F}_\beta}{\beta \tilde{F}_\beta},$$

where $\tilde{z}_u = (1-\beta)m_u + \beta m_{\bar{u}}$. Thus we have shown the following Proposition:

Proposition 2 *Let the weights $w(\beta)$ be given by:*

$$w(\beta)_i = \begin{cases} \frac{1-\beta\hat{F}_\beta}{\beta\hat{F}_\beta} p(y_i \mid x_i, \hat{\lambda}_\beta) & \text{if } y_i = u \\ p(y_i \mid x_i, \hat{\lambda}_\beta) & \text{if } y_i \in \bar{u} \end{cases}, \tag{11}$$

where $\hat{F}_\beta = \tilde{F}_\beta(\hat{\lambda}_\beta)$ denotes the optimal expected F_β measure, that is the one calculated with the model parameters $\lambda = \hat{\lambda}_\beta = argmax\tilde{F}_\beta(\lambda)$. Then, in the notation of Proposition 1

$$\hat{\lambda}_\beta = \hat{\lambda}_{ML}^{w(\beta)},$$

that is $\hat{\lambda}_\beta$ maximizes the $w(\beta)$-weighted log-likelihood as well.

If we additionally assume that the model probabilities in the learned model $p(y_i \mid x_i, \hat{\lambda}_\beta)$ are close to 1 then we can approximate the weights $w(\beta)_i$ from (11) by the following class-wise versions:

$$\bar{w}(\beta)_i = \begin{cases} \frac{1-\beta\tilde{F}_\beta}{\beta\tilde{F}_\beta} & \text{if } y_i = u \\ 1 & \text{if } y_i \in \bar{u} \end{cases}.$$

Hence the following Corollary:

Corollary 1 *If the conditional probabilities $p(y_i \mid x_i, \hat{\lambda}_\beta) \approx 1$ for all i then the weights $w(\beta)$ from (11) can be approximated via class-wise weights $\bar{w}(\beta)$ given by (12).*

$$\bar{w}(\beta)_i = \begin{cases} \frac{1-\beta\hat{F}_\beta}{\beta\hat{F}_\beta} & \text{if } y_i = u \\ 1 & \text{if } y_i \in \bar{u} \end{cases}, \tag{12}$$

where $\hat{F}_\beta$ is as in Proposition 2.

6 Algorithm

The important take away from Sect. 4 is that each maximizer of the expected F_β measure can also be realized as a weighted maximum likelihood. This is very appealing since the log-likelihood is a well behaved concave function. Furthermore, it is important to notice that Corollary 1 and Proposition 2 suggest that we can fall back to weights depending on a single parameter and the qualitative behavior is obvious: larger weights w correspond to smaller β and vice versa.

The expressions (11) for the weights as well as the class-wise approximation (12) are at first sight useless as they involve $\hat{\lambda}_\beta$, i.e. precisely the parameter values we are looking for. However, we can determine them in an adaptive manner: at each iteration perform one full batch step of the gradient ascent optimization of the weighted maximum likelihood with weights as given in Proposition 2 (or Corollary 1 in the class-wise case) but instead of using the unknown optimal $\hat{F}_\beta$ and the conditional model probabilities $p(y_i \mid x_i, \hat{\lambda}_\beta)$ for the optimal parameter $\hat{\lambda}_\beta$ use $\tilde{F}_\beta$ and the model probabilities as achieved at the previous epoch.The Algorithm 1 lists the precise steps.

Algorithm 1 $\tilde{F}_\beta$ maximizer

```
1: $n = 1$
2: Initialize the model parameters and calculate the initial $\tilde{F}_\beta(1)$ from the initialized model.
3: Set the weights (instance-wise)
       $(w_n)_i = \begin{cases} \frac{1-\beta\hat{F}_\beta(n)}{\beta\hat{F}_\beta(n)} p(y_i \mid x_i, \hat{\lambda}_n) & \text{if } y_i = u \\ p(y_i \mid x_i, \hat{\lambda}_n) & \text{if } y_i = \bar{u} \end{cases}$
   or in the class-wise case
       $(\bar{w}_n)_i = \begin{cases} \frac{1-\beta\hat{F}_\beta(n)}{\beta\hat{F}_\beta(n)} & \text{if } y_i = u \\ 1 & \text{if } y_i = \bar{u} \end{cases}$
4: Do one full-batch update of the weighted log-likelihood maximization with weights $w_n$ (respectively
       $\bar{w}(n)$). n:=n+1.
5: Calculate the model expected $F_\beta(n)$ measure and the model conditional probabilities $p(y_i \mid x_i, \hat{\lambda}_n)$.
6: If convergence criteria not met, where convergence means no significant improvement of the target $\tilde{F}_\beta$ in
       the last $k$ steps $\rightarrow$ Step 3.
```

One of the strengths of this algorithm is that it is straightforward to implement. The weighted log-likelihood maximization is analogous to the standard one with a different gradient involving the weights. The implementation then follows some off-the-shelf version of the gradient ascent algorithm.

Convergence of the algorithm: In the following we will show that, given the learning rate for the full-batch gradient ascent is small enough, each step of the instance-wise version of the above algorithm improves the attained value of the expected F_β measure. Hence, if the learning rate is decreasing appropriately the algorithm will converge.

At the n-th step of the (instance-wise) algorithm the following update of the parameters λ is performed:

$$\hat{\lambda}_{n+1} = \hat{\lambda}_{n+1} + \epsilon_n \cdot \nabla_\lambda l^W(\lambda_n : w_n, x, y), \tag{13}$$

where w_n is as described in the algorithm and ϵ_n is the learning rate. Let us define the unscaled weight-functions:

$$W_i(\lambda) = p(y_i \mid x_i, \lambda)\left[\partial_{\tilde{A}}\tilde{F}_\beta(\lambda)\mathbb{1}_{\{y_i=u\}} + \partial_{\tilde{D}}\tilde{F}_\beta(\lambda)\mathbb{1}_{\{y_i=\bar{u}\}}\right].$$

Observe that the weights w_n are simply $W(\hat{\lambda}_n)\big/\partial_{\tilde{D}}\tilde{F}_\beta(\hat{\lambda}_n)$.

For the gradient $\nabla_\lambda l^W(\lambda : w_n, x, y)$ using the definition of w_n we have:

$$\begin{aligned}\nabla_\lambda l^W(\lambda : w_n, x, y) &= \sum_{i=1}^{m}(w_n)_i \frac{1}{p(y_i \mid x_i, \lambda)}\nabla_\lambda p(y_i \mid x_i, \lambda) \\ &= \frac{1}{\partial_{\tilde{D}}\tilde{F}_\beta(\hat{\lambda}_n)}\sum_{i=1}^{m}\frac{p(y_i \mid x_i, \hat{\lambda}_n)}{p(y_i \mid x_i, \lambda)} W_i(\hat{\lambda}_n)\,\nabla_\lambda p(y_i \mid x_i, \lambda).\end{aligned}$$

Now evaluating the above gradient at the current state $\hat{\lambda}_n$ obviously yields:

$$\nabla_\lambda l^W(\lambda : w_n, x, y)\big|_{\lambda=\hat{\lambda}_n} = \frac{1}{\partial_{\tilde{D}} \tilde{F}_\beta(\hat{\lambda}_n)} \sum_{i=1}^{m} W_i(\hat{\lambda}_n) \nabla_\lambda p(y_i \mid x_i, \hat{\lambda}_n)$$
$$= \frac{1}{\partial_{\tilde{D}} \tilde{F}_\beta(\hat{\lambda}_n)} \nabla_\lambda \tilde{F}_\beta(\hat{\lambda}_n). \tag{14}$$

This shows that the calculated gradient of the weighted likelihood is collinear with the gradient of the expected F_β measure (observe that $\partial_{\tilde{D}} \tilde{F}_\beta$ is strictly positive as long as $\beta > 0$). Therefore the update (13) results in an improvement of the expected F_β objective. This proves that the algorithm will eventually converge to a local maximum of the expected F_β.

In a future work the authors intend to investigate scalable strategies to improve Algorithm 1 and to explore the parameter set for the global maximum. Here we briefly sketch two such strategies but we emphasize again that these are merely ideas for future research:

1. One approach would be to consider also versions of the above algorithm where at the n-th iteration $k_n \geq 1$ steps of the gradient ascent algorithm for the weighted log-likelihood function are performed, rather than just a single one as presented in Step 4 of Algorithm 1. As long as $k_n = 1$ is satisfied for all n greater than some large N we still get convergence. However the previous phases with $k_n > 1$ might result in escaping regions with suboptimal local maxima of $\tilde{F}_\beta$. While this is a topic for further research the performed numerical experiments did indicate that the Algorithm 1 tends to converge faster if $k_n > 1$ at least during an initial burn-in phase. Furthermore, our experiments also suggested that the convergence of the algorithm is improved by replacing the expected F_β measure in the expression for the weights w_n ($\bar{w}_n$ respectively) in Step 3 of the Algorithm with the true F_β measure.
2. Another approach which we call *"targeting"* is as follows: One fixes a rather large target value T for the achieved $\tilde{F}_\beta(\hat{\lambda}_\beta)$ and in a burn-in phase runs the algorithm with this value for $\tilde{F}_\beta(\hat{\lambda}_\beta)$, that is use $\tilde{F}_\beta(\hat{\lambda}_\beta) = T$ instead of using the proxy $\tilde{F}_\beta(\hat{\lambda}_\beta) \approx \tilde{F}_\beta(\hat{\lambda}_n)$. Following this phase the algorithm continues with the realized proxy $\tilde{F}_\beta(\hat{\lambda}_n)$ as described in Algorithm 1. The idea behind the *"targeting"* is obvious - we prescribe a target value T for the $\tilde{F}_\beta(\hat{\lambda}_\beta)$ and tries to achieve it by fixing the weights to $(w_n)_i = \frac{1-T}{T} p(y_i \mid x_i, \hat{\lambda}_n) \mathbb{1}_{\{y_i=u\}} + p(y_i \mid x_i, \hat{\lambda}_n) \mathbb{1}_{\{y_i \neq u\}}$.

Clearly an attempt can be made to also merge these two approaches in an appropriate manner. Exploring the targeting strategy as well as variable schedules for k_n and using the true F_β instead of the expected one is in our view a very interesting topic for further research.

7 Baseline methods

We now proceed with the numerical evaluation of our results. We will use two baseline methods in our experiments, with two different goals. First, in order to demonstrate that the algorithm presented in Sect. 6 identifies appropriate weights for the weighted log-likelihood leading to the true optimal $\tilde{F}_\beta$ solution, we compare to a class-wise *brute force* approach which tries a large number of weights (50 between 0 and 1) of the target class and assigns a value of 1 to the other class. Then, for each β, the brute force baseline simply picks the particular weight value which results in the maximum F_β. The *brute-force* approach thus performs an exhaustive search for the optimal weight for the target class within some predefined range $[w_{min}, w_{max}]$. This baseline finds the weights for which the achieved expected F_β is

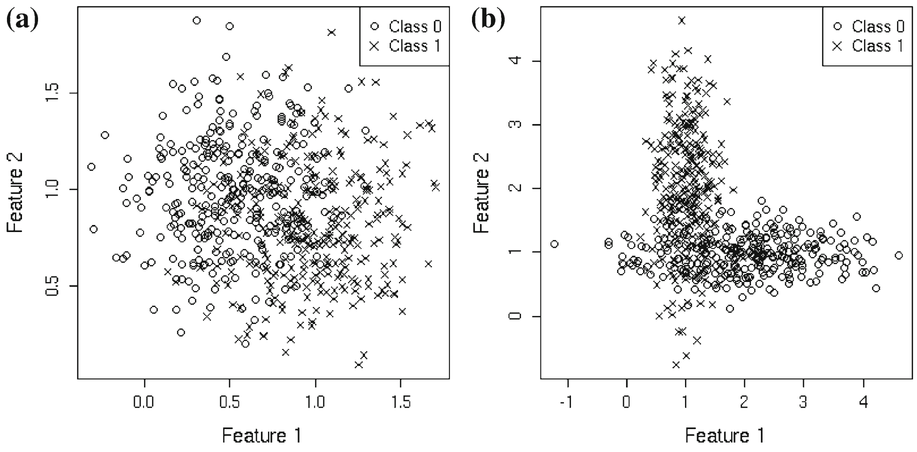

Fig. 1 Distribution of the samples in the space of features for the synthetic dataset: **a** synthetic data A; **b** synthetic data B

optimal and we demonstrate that the class-wise version of our algorithm achieves the same results, however without the computationally extremely exhaustive brute force calculations. We compare our algorithm to this baseline on the train set, as the model is trained on this set and the objective of this comparison is to show that our algorithm fits the training set as well as the brute force approach but with much less computational effort.

The purpose of the second baseline is to compare the performance of our method to the popular approach for adjusting Precision and Recall based on varying the acceptance threshold of a simple maximum entropy model. We call this baseline *acceptance threshold*. The method modifies acceptance thresholds (w.r.t. target class) for the posterior probabilities in order to achieve higher precision (larger threshold) or recall (smaller threshold). For a given β, the probability threshold that gives the best F_β is estimated on the train set and then the threshold is used for prediction on the test set. We use this baseline for comparison on the test set.

8 Data

We tested our algorithm on binary classification tasks on three different datasets, as follows:

Synthetic data - A:

We simulated a dataset of 5000 samples with two classes and two features. Each class contains 2500 samples, distributed as spherical Gaussians in the space of features. The samples from class $\bar{u}$ are distributed as $\mathcal{N}(\mu_0, \Sigma_0)$, with $\mu_0 = (0.5, 1)$ and $\Sigma_0 = (0.3, 0.3)^\top I_2$. Class u is generated by $\mathcal{N}(\mu_1, \Sigma_1)$, with $\mu_1 = (1, 0.8)$ and $\Sigma_1 = (0.3, 0.3)^\top I_2$. In Fig. 1a we visualize the synthetic data A using 600 of its elements. We used 4500 of the samples for training and 500 for testing.

Synthetic data - B:

We simulated a dataset of 5000 samples with two classes and two features. Each class contains 2500 samples, distributed as elliptical Gaussians in the space of features. The samples from class $\bar{u}$ are distributed as $\mathcal{N}(\mu_0, \Sigma_0)$, with $\mu_0 = (2, 1)$ and $\Sigma_0 = (1, 0.3)^\top I_2$. Class u is generated by $\mathcal{N}(\mu_1, \Sigma_1)$, with $\mu_1 = (1, 2)$ and $\Sigma_1 = (0.3, 1)^\top I_2$. In Fig. 1b we visualize the synthetic data B, again using a subset. We used 4500 of the samples for training and 500 for testing.

Sanders Twitter sentiment corpus:

The Sanders Twitter sentiment corpus is free of charge public data set[1] for training and testing sentiment analysis algorithms (Saif et al. 2013) on tweets. It consists of 5513 hand-classified tweets with respect to four different topics "Apple", "Microsoft", "Google" and "Twitter" along with the respective sentiment with regard the tweet's topic - "positive", "neutral", "negative", or "irrelevant". The data set consists of: tweet text; tweet creation date; hand-curated tweet topic; hand-curated sentiment label.

In this work we ignore the topic and focus on tweets expressing an attitude (positive and negative - class 1) vs. impersonal (neutral and irrelevant - class 2). This splits the dataset into 1224 tweets of class 1 that is the class of interest to us (class u in the notation in the theoretical part of the paper) and 4289 of class 2. The distribution of tweets with respect topic and sentiment classes can be found in the set's download package.

In our experiments we started with splitting tweets in tokens; then we use stemming (Porter stemmer) and filtered out the stop words. We conducted some other normalization of the tweets like: all explicit URLs and emails are rewritten as URL and email features. The @"words" and #"words" are rewritten as @tag and #tag and "n't" as "not". Three and more "!" marks are rewritten as "STRONGEST", two as "STRONGER" a single one as "STRONG", while four and more "?" marks are rewritten as "QQQQ+", three as "QQQ", respectively two and one as "QQ" and "Q". We normalized the text to lower case, removed all non-alpha-numeric characters and conjunct tokens up to three at the end of the pipeline. We end up with more than 45000 features from which we selected 50 via evaluation of the information gain of a feature with respect to the classes as in Yang and Pedersen (1997).

The corpus is automatically shuffled before each experiment—90 % of all tweets are used for training and 10 % for testing. The results with this dataset are an average of 10 experiments. The following is a list of the top 10 features for Sanders after the feature selection procedure. (obscene words were removed):

1. QQ 2. STRONG 3. @tag 4. iphone 5. URL 6. ios 7. i_m 8. URL_#tag 9. customer 10. love.

Apple Twitter corpus:

The corpus consists of freely available tweets[2] that mention the word "apple". The corpus is created with the idea to distinguish tweets that discuss "Apple, Inc." (class 1) from tweets about "apple pie" and "apple juice" (class 2). All tweets are 2000; 1306 of which are class 1 and 694 are class 2.

Tweets are tokenized, stemmed and stop words are filtered out. We

and email features. Again the @"words" and #"words" are rewritten as @tag and #tag. Finally we transform the text to lower case, removed all non-alpha-numeric characters and conjunct up to three tokens at the end of the transformation pipeline.

1 http://www.sananalytics.com/lab/twitter-sentiment/.

2 https://dl.dropboxusercontent.com/u/3942841/appleBinaryFiltered.txt.

After these transformations we observe 18462 features from which we selected 276 via the same feature extraction routine as for the Sander twitter corpus. Below is a list of the top 10 features for the Apple Twitter corpus after the feature selection procedure. (obscene words are again removed): 1. iphone 2. apple_juice 3. mac 4. ipod 5. ipad 6. apple_pie 7. iphone_5 8. apple_cider 9. cider 10. patent.

Press association (PA) data:

The "PA" dataset was developed by the Press Association[3] to enable the implementation of a system for recognition and semantic disambiguation of named entities in press releases. Given certain metadata for a number of overlapping candidate entities, an array of features derived from the textual context of their occurrence, and additional document-level metadata, the model is trained to recognize which (if any) of the candidate entities is the one referenced in the text.

The corpus is annotated with respect to people, organization and location mentions; a special "negative" label denotes the candidates that can be considered irrelevant in the given context. We remove non-location entities, thus reducing the problem to a binary classification task, and conduct feature selection to keep $\sim$10 % of the originally extracted features.We split the corpus into a training set (2369 documents) and a held-out test set (160 documents). As a result of this preprocessing, we have 2 classes ("Location" and "Negative"), 48773 instances, a target to irrelevant instance counts ratio of 0.17, and 4640 features.

9 Experiments

On Figs. 2a and b we compare the two versions of our algorithm to the *brute-force* baseline which finds the optimal solution using an exhaustive search which is computationally very demanding. The objective is to demonstrate that our algorithm does find the optimal solution with far less iterations. The results clearly demonstrate that on the synthetic datasets (train) our *class-wise* version of the algorithm is indeed very close to the *brute-force* baseline. However, the brute-force method requires training 50 weighted maximum entropy models, whereas our approach needs only one.The *instance-wise* approach ensures a better adaptation to the data and therefore achieves even higher F_β values than the class-wise *brute-force*.Additionally we have added the F_β as achieved by the standard rigid maximum entropy model to demonstrate the scale of improvement that can be achieved with a dedicated F_β optimizer. The comparison to the standard Maxent clearly underlines the importance of the targeted F_β optimization.

Figures 3a and b depict the performance of the two versions of our algorithm (instance-wise and class-wise) on the test subset of the synthetic datasets A and B, respectively, compared with the acceptance threshold baseline. On the synthetic dataset A, the F_β values are comparable. We believe in fact that for the two spherical Gaussian distributions, the acceptance threshold should be nearly optimal, as translations of the maximum likelihood model to the left and right are the best choice for trading Precision and Recall. We elaborate on this in the next section. On the other hand, when the class distributions are skewed as in the synthetic data B, our algorithm is clearly superior.

As noted above we have tested our algorithm also on two publicly available datasets: the Sanders Twitter corpus and the Apple Twitter corpus. The results on the train and test subsets of the Sanders Twitter dataset are shown on Fig. 4.

[3] http://www.pressassociation.com/.

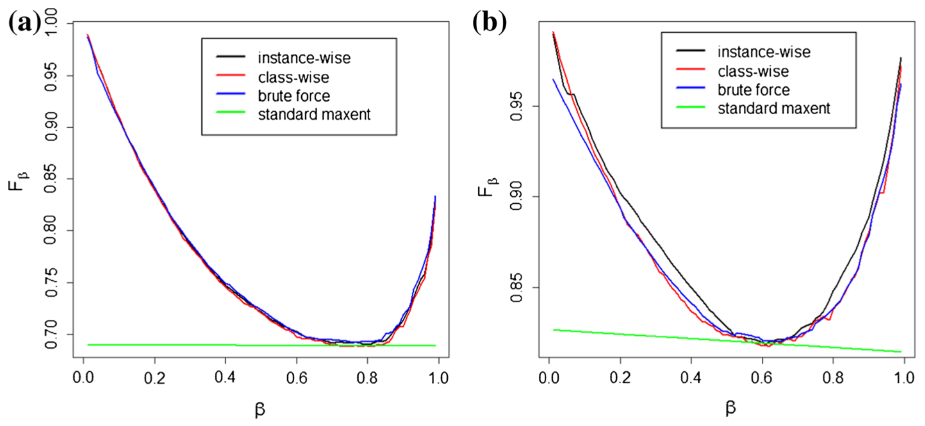

Fig. 2 Comparison between the F_β achieved by the brute force baseline and the class-wise and the instance-wise versions of our algorithm on the train subset **a** synthetic data A; **b** synthetic data B

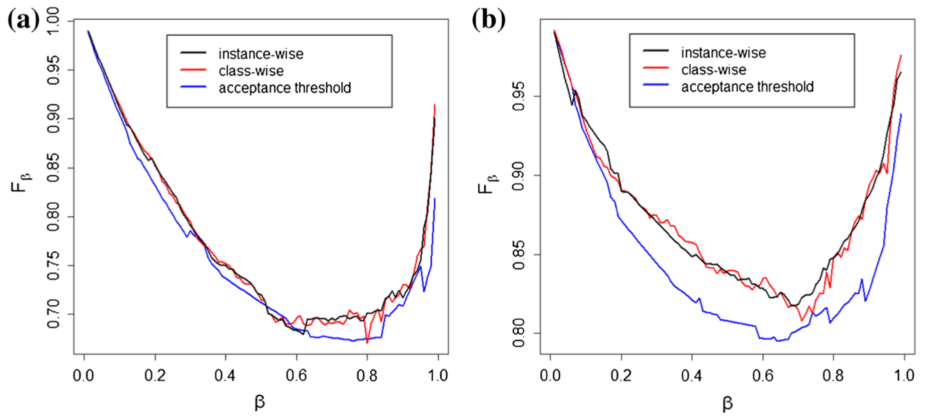

Fig. 3 Comparison between the F_β achieved by the acceptance threshold baseline, the class-wise and the instance-wise versions of our algorithm on the test subset **a** synthetic data A; **b** synthetic data B

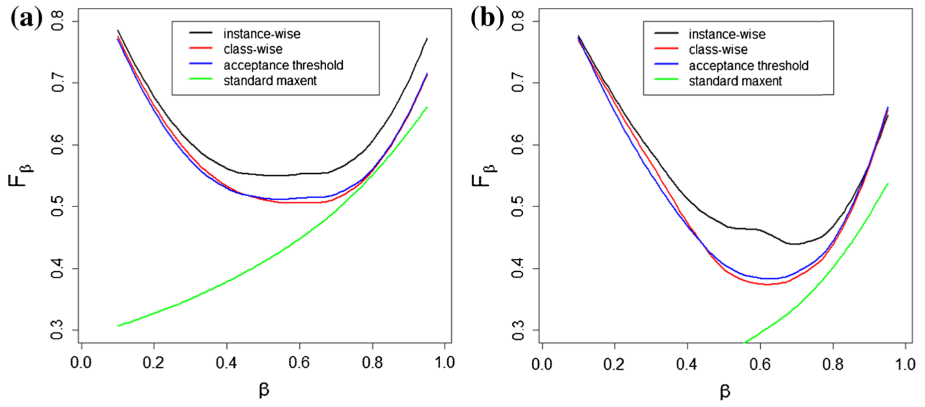

Fig. 4 Comparison of the F_β achieved by the rigid Maxent, the acceptance threshold baseline, the class-wise and the instance-wise versions of our algorithm on the Sanders Twitter corpus **a** train subset **b** test subset

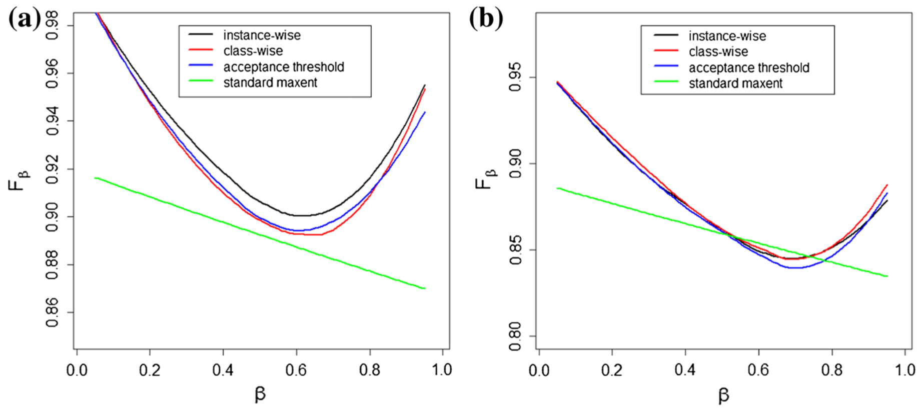

Fig. 5 Comparison of the F_β achieved by the rigid Maxent, the acceptance threshold baseline, the class-wise and the instance-wise versions of our algorithm on the Apple Twitter corpus **a** train subset **b** test subset

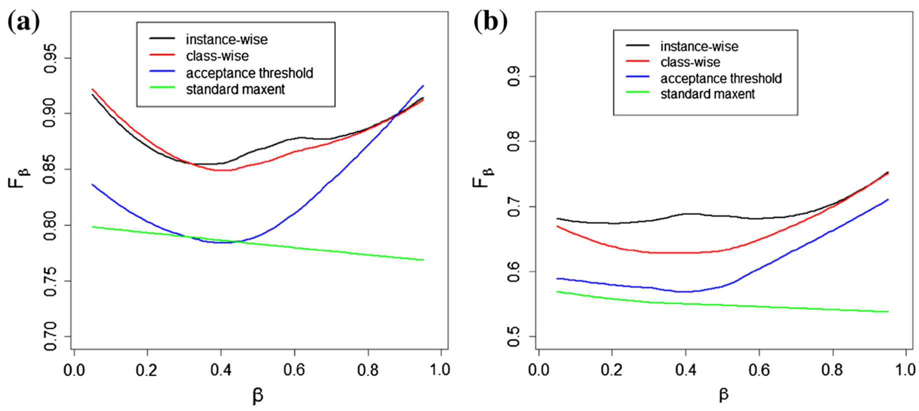

Fig. 6 Comparison of the algorithms on the Press Association data set **a** train subset **b** test subset

The instance-wise version of the algorithm achieves a significant improvement of the F_β measure compared to the acceptance threshold baseline. The improvement of the class-wise version is however marginal.

While the improvement on the Apple Twitter corpus (Fig. 5) is not that impressive it is still visible and given the rather large values of the achieved F_β not to be underestimated as well. Clearly, the Apple Twitter corpus (Fig. 5) is more of the "Spherical type" rather than the "Orthogonal type" (see the discussion in Sect. 10).

Finally, Fig. 6 shows the performance of our algorithm and the acceptance threshold baseline on the Press Association dataset. Clearly both the class-wise and instance-wise algorithms outperform the acceptance threshold baseline by a generous margin. In particular, the instance-wise approach is visibly better than the class-wise one.

Following the observations on the two simulated datasets, we believe that the Press Association and the Sanders Twitter data, and perhaps a good portion of the NLP data in general, are distributed in a non-trivial way in the space of features, such that adaptive approaches for Precision-Recall trade-off like the one we are proposing are indeed useful. Clearly, as the experiments on the Apple Twitter corpus show, the improvement over the acceptance

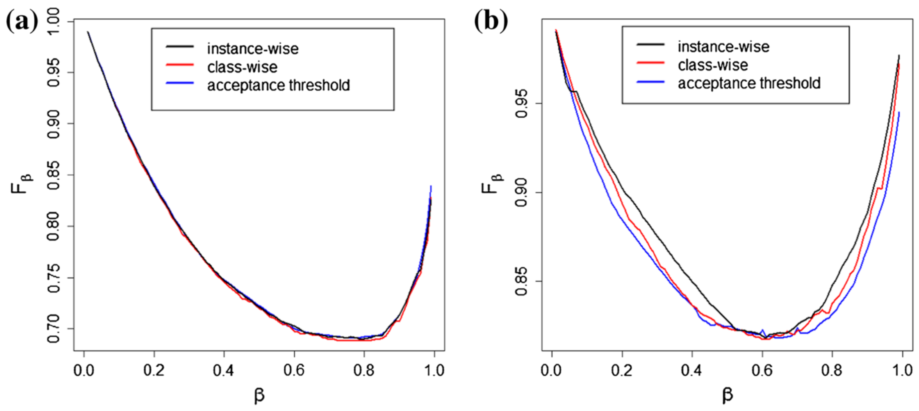

Fig. 7 Comparison between the F_β achieved by the acceptance threshold baseline and the class-wise and the instance-wise versions of our algorithm (train set) **a** synthetic data A; **b** synthetic data B

threshold depends on the dataset as well as on the used features and might not be always substantial but given that our approach comes at virtually no additional computational and implementation cost and is guaranteed at least not to underperform compared to the adjustment threshold (on the training dataset) we believe that it is a good choice for F_β optimization.

10 Limits and merits of the weighted maximum entropy

In this section we compare the class-wise weighted maximum entropy and the acceptance threshold method with the help of the two stylized artificial data sets A and B shown on Fig. 1.

The acceptance threshold corresponds to a translation of the separating hyperplane obtained by the standard maximum entropy model. It is rather clear that with translation we can achieve an optimal Precision/Recall trade-off for the synthetic dataset A. Indeed on Fig. 3a one sees that the acceptance threshold and the weighted maximum entropy do result in similar optimal F_β values on the test set. On the train set the acceptance threshold and our algorithms generate virtually the same optimal F_β values.

Observe that Fig. 7 merely reiterates on the *train set* the comparisons from Fig. 3.

The optimal Precision/Recall trade-off for the synthetic dataset B however requires additional rotation/tilting of the separating hyperplane that cannot be produced by adjusting the acceptance threshold. In line with this intuition Figs. 3b and 7b, for the test and train sets respectively, demonstrate that the weighted likelihood settles at a considerably better Precision-Recall pairs and consequently results in larger F_β values on the test and the train sets.

Clearly, in the general case the optimal shift of the separating plane is expected to have a rotation component that is unaccessible by simply adjusting the acceptance threshold. Moreover as shown in the previous section the instance-wise weighted maximum entropy also outperforms the class-wise approximation.

11 Conclusions and future work

The main result of the paper is that the weighted maximum likelihood and the expected F_β measure are simply two different ways to specify a particular trade-off between the objectives of the same MOP. As a consequence, each expected F_β maximizer can be realized as a weighted maximum likelihood estimator and approximated via a class-wise weighted maximum likelihood estimator.

The difficulty in exploiting the statement of Proposition 1 lies in the fact that it is not a priori clear how to choose the weights $w(\beta)$ for a given β.

We presented a theoretical result giving the optimal $w(\beta)$ as well as an efficient algorithm for maximizing the $\tilde{F}_\beta$ by adaptively determining the right weights. We have tested the algorithm on various data sets and the experiments suggest that it indeed finds the optimal weights and achieves $\tilde{F}_\beta$ maximization. Furthermore we have compared the algorithm to the optimal threshold baseline showing the superiority of our approach.

The presented results can be generalized to the regularized and multi-class cases and will be presented in a forthcoming article. Lastly, the proposed approach to view a broad class of probabilistic learning schemes based on optimizing some objective function as a specific trade-off between the underlying conditional probabilities and thus to link it to a weighted maximum likelihood model can be applied beyond the specific set-up of maximum entropy and the F_β objective. In particular, the intuition via modifying the original data set and then using the standard model on the new data set, as explained above, can obviously be applied to a very broad range of machine learning models.

Acknowledgments The work, described in this paper, is supported by the FP7-ICT Strategic Targeted Research Project PHEME (No. 611233).

References

Berger, A., Della Pietra, V., & Della Pietra, S. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

Carpenter, B. (2007). Lingpipe for 99.99 % recall of gene mentions. In: Proceedings of the 2nd BioCreative, workshop.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Culotta, A. (2004). Confidence estimation for information extraction. In: Proceedings of Human language technology conference and North American chapter of the Association for Computational Linguistics (HLT-NAACL).

Dembczyn'ski, K., Waegeman, W., Cheng, W., Hü llermeier, E. (2011). An exact algorithm for f-measure maximization. In: Neural information processing systems : 2011 conference book. Neural Information Processing Systems Foundation.

Ehrgott, M. (2005). *Multi criteria optimization*. New Jersery: Springer.

Ganchev, K., Crammer, K., Pereira, F., Mann, G., Bellare, K., Mccallum, A., Carroll, S., Jin, Y., White, P. (2007). Penn/umass/chop biocreative ii systems 1 penn/umass/chop biocreative ii systems. In: Proceedings of the second bioCreative challenge evaluation workshop.

Ganchev, K., Pereira, O., Mandel, M., Carroll, S., White, P. (2007). Semi-automated named entity annotation. In: Proceedings of the linguistic annotation workshop, Prague, Czech Republic. Association for, Computational Linguistics.

Geoffrion, A. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, *22*, 618–630.

Georgiev, G., Ganchev, K., Momtchev, V., Peychev, D., Nakov, P., Roberts, A. (2009). Tunable domain-independent event extraction in the mira framework. In: Proceedings of the workshop on current trends in biomedical natural language processing: Shared Task, BioNLP '09, pp. 95–98.

Jansche, M. (2005).Maximum expected F-measure training of logistic regression models. In: HLT '05, Association for computational linguistics, Morristown, NJ, USA, pp. 692–699

Joachims, T. (2005).A support vector method for multivariate performance measures. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 377–384. ACM Press.

Klinger, R., Friedrich, C.M. (2009). User's choice of precision and recall in named entity recognition. In: Proceedings of the International Conference RANLP-2009, pp. 192–196.

Lafferty, J. (2001).Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pp. 282–289.

Minkov, E., Wang, R., Tomasic, A., Cohen, W. (2006). NER systems that suit user’s preferences: adjusting the recall-precision trade-off for entity extraction. In: Proceedings of NAACL, pp. 93–96.

Nan, Y., Chai, K.M.A., Lee, W.S., Chieu, H.L. (2012). Optimizing f-measure: A tale of two approaches. In: ICML. http://dblp.uni-trier.de/db/conf/icml/icml2012.html#NanCLC12

Saif, H., Fernandez, M., He, Y., Alani, H. (2013). Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold.

Simecková, M. (2005). Maximum weighted likelihood estimator in logistic regression.

Suzuki, J., McDermott, E., Isozaki, H. (2006). Training conditional random fields with multivariate evaluation measures. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for, Computational Linguistics, ACL-44, pp. 217–224.

Vandev, D.L., Neykov, N.M. (1998). About regression estimators with high breakdown point. Statistics 32, 111–129. http://www.informaworld.com/10.1080/02331889808802657.

Yang, Y., Pedersen, J.O. (1997). A comparative study on feature selection in text categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning, ICML ’97, pp. 412–420. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. http://dl.acm.org/citation.cfm?id=645526.657137