Maximum Expected F-Measure Training of Logistic Regression Models

Martin Jansche
Center for Computational Learning Systems
Columbia University
New York, NY 10027, USA
jansche@acm.org

Abstract

We consider the problem of training logistic regression models for binary classification in information extraction and information retrieval tasks. Fitting probabilistic models for use with such tasks should take into account the demands of the task-specific utility function, in this case the well-known F-measure, which combines recall and precision into a global measure of utility. We develop a training procedure based on empirical risk minimization / utility maximization and evaluate it on a simple extraction task.

1 Introduction

Log-linear models have been used in many areas of Natural Language Processing (NLP) and Information Retrieval (IR). Scenarios in which log-linear models have been applied often involve simple binary classification decisions or probability assignments, as in the following three examples: Ratnaparkhi et al. (1994) consider a restricted form of the prepositional phrase attachment problem where attachment decisions are binary; Ittycheriah et al. (2003) reduce entity mention tracking to the problem of modeling the probability of two mentions being linked; and Greiff and Ponte (2000) develop models of probabilistic information retrieval that involve binary decisions of relevance. What is common to all three approaches is the application of log-linear models to binary classification tasks.[1] As Ratnaparkhi (1998, p. 27f.) points out, log-linear models of binary response variables are equivalent to, and in fact mere notational variants of, logistic regression models.

In this paper we focus on binary classification tasks, and in particular on the loss or utility associated with classification decisions. The three problems mentioned before – prepositional phrase attachment, entity mention linkage, and relevance of a document to a query – differ in one crucial aspect: The first is evaluated in terms of accuracy or, equivalently, symmetric zero–one loss; but the second and third are treated as information extraction/retrieval problems and evaluated in terms of recall and precision. Recall and precision are combined into a single overall utility function, the well-known F-measure. It may be desirable to estimate the parameters of a logistic regression model by maximizing F-measure during training. This is analogous, and in a certain sense equivalent, to empirical risk minimization, which has been used successfully in related areas, such as speech recognition (Rahim and Lee, 1997), language modeling (Paciorek and Rosenfeld, 2000), and machine translation (Och, 2003).

The novel contribution of this paper is a training procedure for (approximately) maximizing the expected F-measure of a probabilistic classifier based on a logistic regression model. We formulate a vector-valued utility function which has a well-defined expected value; F-measure is then a rational function of this expectation and can be maximized numerically under certain conventional regularizing assumptions.

[1] These kinds of log-linear models are also known among the NLP community as "maximum entropy models" (Berger et al., 1996; Ratnaparkhi, 1998). This is an unfortunate choice of terminology, because the term "maximum entropy" does not uniquely determine a family of models unless the constraints subject to which entropy is being maximized are specified.

We begin with a review of logistic regression (Section 2) and then discuss the use of F-measure for evaluation (Section 3). We reformulate F-measure as a function of an expected utility (Section 4) which is maximized during training (Section 5). We discuss the differences between our parameter estimation technique and maximum likelihood training on a toy example (Section 6) as well as on a real extraction task (Section 7). We conclude with a discussion of further applications and generalizations (Section 8).

2 Review of Logistic Regression

Bernoulli regression models are conditional probability models of a binary response variable Y given a vector $\vec{X}$ of k explanatory variables $(X_1, \ldots, X_k)$. We will use the convention[2] that Y takes on a value $y \in \{-1, +1\}$.

Logistic regression models (Cox, 1958) are perhaps best viewed as instances of generalized linear models (Nelder and Wedderburn, 1972; McCullagh and Nelder, 1989) where the the response variable follows a Bernoulli distribution and the link function is the logit function. Let us summarize this first, before expanding the relevant definitions:

$$Y \sim \text{Bernoulli}(p)$$
$$\text{logit}(p) = \theta_0 + x_1\,\theta_1 + x_2\,\theta_2 + \cdots + x_k\,\theta_k$$

What this means is that there is an unobserved quantity p, the success probability of the Bernoulli distribution, which we interpret as the probability that Y will take on the value $+1$:

$$\Pr(Y = +1 \,|\, \vec{X} = (x_1, x_2, \ldots, x_k), \vec{\theta}) = p$$

The logit (log odds) function is defined as follows:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

The logit function is used to transform a probability, constrained to fall within the interval $(0,1)$, into a real number ranging over $(-\infty, +\infty)$. The inverse function of the logit is the cumulative distribution function of the standard logistic distribution (also known as the *sigmoid* or *logistic* function), which we call g:

$$g(z) = \frac{1}{1+\exp(-z)}$$

This allows us to write

$$p = g(\theta_0 + x_1\,\theta_1 + x_2\,\theta_2 + \cdots + x_k\,\theta_k).$$

We also adopt the usual convention that $\vec{x} = (1, x_1, x_2, \ldots, x_k)$, which is a $k+1$-dimensional vector whose first component is always 1 and whose remaining k components are the values of the k explanatory variables. So the Bernoulli probability can be expressed as

$$p = g\left(\sum_{j=0}^{k} x_j\,\theta_j\right) = g\left(\vec{x}\cdot\vec{\theta}\right).$$

The conditional probability model then takes the following abbreviated form, which will be used throughout the rest of this paper:

$$\Pr(+1 \,|\, \vec{x}, \vec{\theta}) = \frac{1}{1+\exp(-\vec{x}\cdot\vec{\theta})} \qquad (1)$$

A classifier can be constructed from this probability model using the MAP decision rule. This means predicting the label $+1$ if $\Pr(+1 \,|\, \vec{x}, \vec{\theta})$ exceeds $1/2$, which amounts to the following:

$$y_{\text{map}}(\vec{x}) = \operatorname*{argmax}_y \Pr(y \,|\, \vec{x}, \vec{\theta}) = \text{sgn}\left(\vec{x}\cdot\vec{\theta}\right)$$

This illustrates the well-known result that a MAP classifier derived from a logistic regression model is equivalent to a (single-layer) perceptron (Rosenblatt, 1958) or linear threshold unit.

3 F-Measure

Suppose the parameter vector θ of a logistic regression model is known. The performance of the resulting classifier can then be evaluated in terms of the *recall* (or *sensitivity*) and *precision* of the classifier on an evaluation dataset. Recall (R) and precision (P) are defined in terms of the number of true positives (A), misses (B), and false alarms (C) of the classifier (cf. Table 1):

$$R = \frac{A}{A+B} \qquad P = \frac{A}{A+C}$$

[2] The natural choice may seem to be for Y to range over the set $\{0,1\}$, but the convention adopted here is more common for classification problems and has certain advantages which will become clear soon.

		predicted		total
		$+1$	-1	
true	$+1$	A	B	n_{pos}
	-1	C	D	n_{neg}
total		m_{pos}	m_{neg}	n

Table 1: Schema for a 2×2 contingency table

The F_α measure – familiar from Information Retrieval – combines recall and precision into a single utility criterion by taking their α-weighted harmonic mean:

$$F_\alpha(R,P) = \left(\alpha\frac{1}{R} + (1-\alpha)\frac{1}{P}\right)^{-1}$$

The F_α measure can be expressed in terms of the triple (A,B,C) as follows:

$$F_\alpha(A,B,C) = \frac{A}{A + \alpha B + (1-\alpha)C} \qquad (2)$$

In order to define A, B, and C formally, we use the notation $[\![\pi]\!]$ to denote a variant of the Kronecker delta defined like this, where π is a Boolean expression:

$$[\![\pi]\!] = \begin{cases} 1 & \text{if } \pi \\ 0 & \text{if } \neg\pi \end{cases}$$

Given an evaluation dataset $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ the counts of hits (true positives), misses, and false alarms are, respectively:

$$A = \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\,[\![y_i = +1]\!]$$
$$B = \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = -1]\!]\,[\![y_i = +1]\!]$$
$$C = \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\,[\![y_i = -1]\!]$$

Note that F-measure is seemingly a global measure of utility that applies to an evaluation dataset as a whole: while the F-measure of a classifier evaluated on a single supervised instance is well defined, the overall F-measure on a larger dataset is not a function of the F-measure evaluated on each instance in the dataset. This is in contrast to ordinary loss/utility, whose grand total (or average) on a dataset can be computed by direct summation.

4 Relation to Expected Utility

We reformulate F-measure as a scalar-valued rational function composed with a vector-valued utility function. This allows us to define notions of expected and average utility, setting up the discussion of parameter estimation in terms of empirical risk minimization (or rather, utility maximization).

Define the following vector-valued utility function u, where $u(\tilde{y}\,|\,y)$ is the utility of choosing the label $\tilde{y}$ if the true label is y:

$$u(+1\,|\,+1) = (1,0,0)$$
$$u(-1\,|\,+1) = (0,1,0)$$
$$u(+1\,|\,-1) = (0,0,1)$$
$$u(-1\,|\,-1) = (0,0,0)$$

This function indicates whether a classification decision is a hit, miss, or false alarm. Correct rejections are not counted.

Expected values are, of course, well-defined for vector-valued functions. For example, the expected utility is

$$\mathrm{E}[u] = \sum_{(\vec{x},y)} u(y_{\text{map}}(\vec{x})\,|\,y)\ \Pr(\vec{x},y).$$

In empirical risk minimization we approximate the expected utility of a classifier by its average utility U_S on a given dataset $S = (\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$:

$$\begin{aligned}\mathrm{E}[u] \approx U_S &= \frac{1}{n}\sum_{i=1}^{n} u(y_{\text{map}}(\vec{x}_i)\,|\,y_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} u(+1\,|\,y_i)\,[\![y_{\text{map}}(\vec{x}_i) = +1]\!] \\ &\qquad + u(-1\,|\,y_i)\,[\![y_{\text{map}}(\vec{x}_i) = -1]\!]\end{aligned}$$

Now it is easy to see that U_S is the following vector:

$$U_S = \frac{1}{n}\begin{pmatrix} \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\,[\![y_i = +1]\!] \\ \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = -1]\!]\,[\![y_i = +1]\!] \\ \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\,[\![y_i = -1]\!] \end{pmatrix} \qquad (3)$$

So $U_S = n^{-1}(A,B,C)$ where A, B, and C are as defined before. This means that we can interpret the

F-measure of a classifier as a simple rational function of its empirical average utility (the scaling factor $1/n$ in (3) can in fact be omitted). This allows us to approach the parameter estimation task as an empirical risk minimization or utility maximization problem.

5 Discriminative Parameter Estimation

In the preceding two sections we assumed that the parameter vector $\vec{\theta}$ was known. Now we turn to the problem of estimating $\vec{\theta}$ by maximizing the F-measure formulated in terms of expected utility. We make the dependence on $\vec{\theta}$ explicit in the formulation of the optimization task:

$$\vec{\theta}^\star = \operatorname*{argmax}_{\vec{\theta}} F_\alpha(A(\vec{\theta}), B(\vec{\theta}), C(\vec{\theta})),$$

where $(A(\vec{\theta}), B(\vec{\theta}), C(\vec{\theta})) = U_S(\vec{\theta})$ as defined in (3). We encounter the usual problem: the basic quantities involved are integers (counts of hits, misses, and false alarms), and the optimization objective is a piecewise-constant functions of the parameter vector $\vec{\theta}$, due to the fact that $\vec{\theta}$ occurs exclusively inside Kronecker deltas. For example:

$$[\![y_{\text{map}}(\vec{x}) = +1]\!] = \Big[\!\Big[\Pr(+1 \,|\, \vec{x}, \vec{\theta}) > 0.5 \Big]\!\Big]$$

In general, we can set

$$\Big[\!\Big[\Pr(+1 \,|\, \vec{x}, \vec{\theta}) > 0.5 \Big]\!\Big] \approx \Pr(+1 \,|\, \vec{x}, \vec{\theta}), \qquad (4)$$

and in the case of logistic regression this arises as a special case of approximating the limit

$$\Big[\!\Big[\Pr(+1 \,|\, \vec{x}, \vec{\theta}) > 0.5 \Big]\!\Big] = \lim_{\gamma \to \infty} g(\gamma\, \vec{x} \cdot \vec{\theta})$$

with a fixed value of $\gamma = 1$. The choice of γ does not matter much. The important point is that we are now dealing with approximate quantities which depend continuously on $\vec{\theta}$. In particular $A(\vec{\theta}) \approx \tilde{A}(\vec{\theta})$, where

$$\tilde{A}(\vec{\theta}) = \sum_{\substack{i=1 \\ y_i = +1}}^{n} g(\gamma\, \vec{x}_i \cdot \vec{\theta}). \qquad (5)$$

Since the marginal total of positive instances n_{pos} (cf. Table 1) does not depend on $\vec{\theta}$, we use the identities $\tilde{B}(\vec{\theta}) = n_{\text{pos}} - \tilde{A}(\vec{\theta})$ and $\tilde{m}_{\text{pos}}(\vec{\theta}) = \tilde{A}(\vec{\theta}) + \tilde{C}(\vec{\theta})$ to rewrite the optimization objective as $\tilde{F}_\alpha$:

$$\tilde{F}_\alpha(\vec{\theta}) = \frac{\tilde{A}(\vec{\theta})}{\alpha\, n_{\text{pos}} + (1-\alpha)\, \tilde{m}_{\text{pos}}(\vec{\theta})}, \qquad (6)$$

where $\tilde{A}(\vec{\theta})$ is given by (5) and $\tilde{m}_{\text{pos}}(\vec{\theta})$ is

$$\tilde{m}_{\text{pos}}(\vec{\theta}) = \sum_{i=1}^{n} g(\gamma\, \vec{x}_i \cdot \vec{\theta}).$$

Maximization of $\tilde{F}$ as defined in (6) can be carried out numerically using multidimensional optimization techniques like conjugate gradient search (Fletcher and Reeves, 1964) or quasi-Newton methods such as the BFGS algorithm (Broyden, 1967; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). This requires the evaluation of partial derivatives. The jth partial derivative of $\tilde{F}$ is as follows:

$$\frac{\partial \tilde{F}(\vec{\theta})}{\partial \theta_j} = h \frac{\partial \tilde{A}(\vec{\theta})}{\partial \theta_j} - h^2\, \tilde{A}(\vec{\theta})\, (1-\alpha) \frac{\partial \tilde{m}_{\text{pos}}(\vec{\theta})}{\partial \theta_j}$$

$$h = \frac{1}{\alpha\, n_{\text{pos}} + (1-\alpha)\, \tilde{m}_{\text{pos}}(\vec{\theta})}$$

$$\frac{\partial \tilde{A}(\vec{\theta})}{\partial \theta_j} = \sum_{\substack{i=1 \\ y_i = +1}}^{n} g'(\gamma\, \vec{x}_i \cdot \vec{\theta})\, \gamma x_{ij}$$

$$\frac{\partial \tilde{m}_{\text{pos}}(\vec{\theta})}{\partial \theta_j} = \sum_{i=1}^{n} g'(\gamma\, \vec{x}_i \cdot \vec{\theta})\, \gamma x_{ij}$$

$$g'(z) = g(z)\, (1 - g(z))$$

One can compute the value of $\tilde{F}(\vec{\theta})$ and its gradient $\nabla \tilde{F}(\vec{\theta})$ simultaneously at a given point $\vec{\theta}$ in $O(nk)$ time and $O(k)$ space. Pseudo-code for such an algorithm appears in Figure 1. In practice, the inner loops on lines 8–9 and 14–18 can be made more efficient by using a sparse representation of the row vectors $x[i]$. A concrete implementation of this algorithm can then be used as a callback to a multidimensional optimization routine. We use the BFGS minimizer provided by the GNU Scientific Library (Galassi et al., 2003). Important caveat: the function $\tilde{F}$ is generally not concave. We deal with this problem by taking the maximum across several runs of the optimization algorithm starting from random initial values. The next section illustrates this point further.

x	y
0	$+1$
1	-1
2	$+1$
3	$+1$

Table 2: Toy dataset

6 Comparison with Maximum Likelihood

A comparison with the method of maximum likelihood illustrates two important properties of discriminative parameter estimation. Consider the toy dataset in Table 2 consisting of four supervised instances with a single explanatory variable. Thus the logistic regression model has two parameters and takes the following form:

$$\Pr_{\text{toy}}(+1 \,|\, x, \theta_0, \theta_1) = \frac{1}{1+\exp(-\theta_0 - x\,\theta_1)}$$

The log-likelihood function L is simply

$$\begin{aligned} L(\theta_0, \theta_1) = &\log \Pr_{\text{toy}}(+1 \,|\, 0, \theta_0, \theta_1) \\ &+ \log \Pr_{\text{toy}}(-1 \,|\, 1, \theta_0, \theta_1) \\ &+ \log \Pr_{\text{toy}}(+1 \,|\, 2, \theta_0, \theta_1) \\ &+ \log \Pr_{\text{toy}}(+1 \,|\, 3, \theta_0, \theta_1). \end{aligned}$$

A surface plot of L is shown in Figure 2. Observe that L is concave; its global maximum occurs near $(\theta_0, \theta_1) \approx (0.35, 0.57)$, and its value is always strictly negative because the toy dataset is not linearly separable. The classifier resulting from maximum likelihood training predicts the label $+1$ for all training instances and thus achieves a recall of $3/3$ and precision $3/4$ on its training data. The $F_{\alpha=0.5}$ measure is $6/7$.

Contrast the shape of the log-likelihood function L with the function $\tilde{F}_\alpha$. Surface plots of $\tilde{F}_{\alpha=0.5}$ and $\tilde{F}_{\alpha=0.25}$ appear in Figure 3. The figures clearly illustrate the first important (but undesirable) property of $\tilde{F}$, namely the lack of concavity. They also illustrate a desirable property, namely the ability to take into account certain properties of the loss function during training. The $\tilde{F}_{\alpha=0.5}$ surface in the left panel of Figure 3 achieves its maximum in the right corner for $(\theta_0, \theta_1) \to (+\infty, +\infty)$. If we choose $(\theta_0, \theta_1) = (20, 15)$ the classifier labels every instance of the training data with $+1$.

fdf(θ):

```
1:  m ← 0
2:  A ← 0
3:  for j ← 0 to k do
4:      dm[j] ← 0
5:      dA[j] ← 0
6:  for i ← 1 to n do
7:      p ← 0
8:      for j ← 0 to k do
9:          p ← p + x[i][j] × θ[j]
10:     p ← 1/(1 + exp(−d))
11:     m ← m + p
12:     if y[i] = +1 then
13:         A ← A + p
14:     for j ← 0 to k do
15:         t ← p × (1 − p) × x[i][j]
16:         dm[j] ← dm[j] + t
17:         if y[i] = +1 then
18:             dA[j] ← dA[j] + t
19: h ← 1/(α × n_pos + (1 − α) × m)
20: F ← h × A
21: t ← F × (1 − α)
22: for j ← 0 to k do
23:     dF[j] ← h × (dA[j] − t × dm[j])
24: return (F, dF)
```

Figure 1: Algorithm for computing $\tilde{F}$ and $\nabla\tilde{F}$

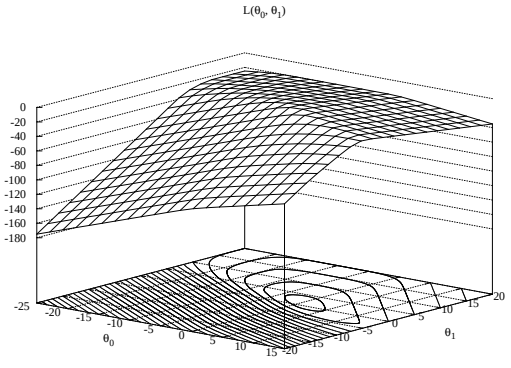

Figure 2: Surface plot of L on the toy dataset

Observe the difference between the $\tilde{F}_{\alpha=0.5}$ surface and the $\tilde{F}_{\alpha=0.25}$ surface in the right hand panel of Figure 3: $\tilde{F}_{\alpha=0.25}$ achieves its maximum in the back corner for $(\theta_0, \theta_1) \to (-\infty, +\infty)$. If we set $(\theta_0, \theta_1) = (-20, 15)$ the resulting classifier labels the first two

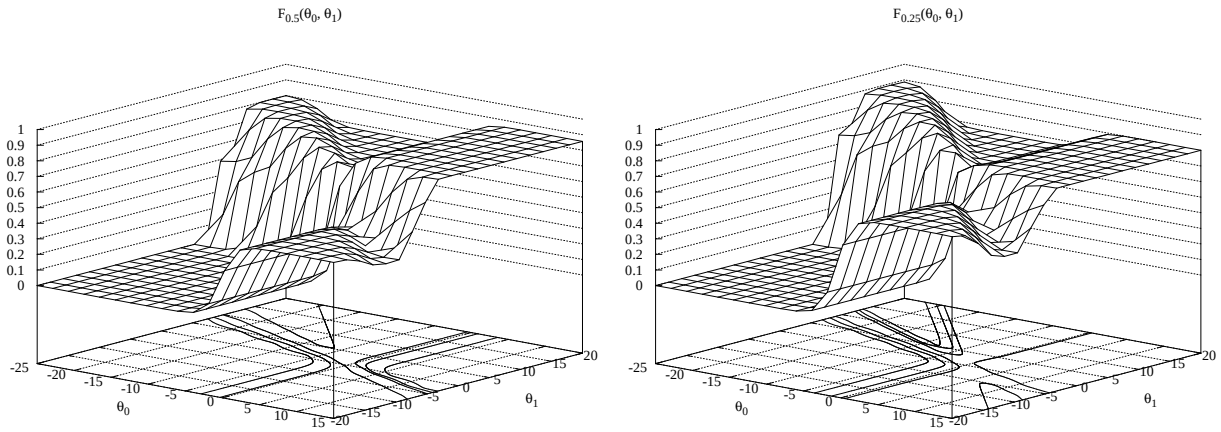

Figure 3: Surface plot of $\tilde{F}_{\alpha=0.5}$ (left) and $\tilde{F}_{\alpha=0.25}$ (right) on the toy dataset

instances ($x = 0$ and $x = 1$) as -1 and the last two instances ($x = 2$ and $x = 3$) as $+1$.

The classifier trained according to the $\tilde{F}_{\alpha=0.5}$ criterion achieves an $F_{\alpha=0.5}$ measure of $6/7 \approx 0.86$, compared with $4/5 = 0.80$ for the classifier trained according to the $\tilde{F}_{\alpha=0.25}$ criterion. Conversely, that classifier achieves an $F_{\alpha=0.25}$ measure of $8/9 \approx 0.89$ compared with $4/5 = 0.80$ for the classifier trained according to the $\tilde{F}_{\alpha=0.5}$ criterion. This demonstrates that the training procedure can effectively take information from the utility function into account, producing a classifier that performs well under a given evaluation criterion. This is the result of optimizing a task-specific utility function during training, not simply a matter of adjusting the decision threshold of a trained classifier.

7 Evaluation on an Extraction Problem

We evaluated our discriminative training procedure on a real extraction problem arising in broadcast news summarization. The overall task is to summarize the stories in an audio news broadcast (or in the audio portion of an A/V broadcast). We assume that story boundaries have been identified and that each story has been broken up into sentence-like units. A simple way of summarizing a story is then to classify each sentence as either belonging into a summary or not, so that a relevant subset of sentences can be extracted to form the basis of a summary. What makes the classification task hard, and therefore interesting, is the fact that reliable features are hard to come by. Existing approaches such as Maskey and Hirschberg 2005 do well only when combining diverse features such as lexical cues, acoustic properties, structural/ positional features, etc.

The task has another property which renders it problematic, and which prompted us to develop the discriminative training procedure described in this paper. Summarization, by definition, aims for brevity. This means that in any dataset the number of positive instances will be much smaller than the number of negative instances. Given enough data, balance could be restored by discarding negative instances. This, however, was not an option in our case: a moderate amount of manually labeled data had been produced and about one third would have had to be discarded to achieve a balance in the distribution of class labels. This would have eliminated precious supervised training data, which we were not prepared to do.

The training and test data were prepared by Maskey and Hirschberg (2005), who performed the feature engineering, imputation of missing values, and the training–test split. We used the data unchanged in order to allow for a comparison between approaches. The dataset is made up of 30 features, divided into one binary response variable, and one binary explanatory variable plus 28 integer- and real-valued explanatory variables. The training portion consists of 3 535 instances, the test portion of 408 instances.

We fitted logistic regression models in three different ways: by maximum likelihood ML, by $\tilde{F}_{\alpha=0.5}$ maximization, and by $\tilde{F}_{\alpha=0.75}$ maximization. Each

Method	R	P	$F_{\alpha=0.5}$	$F_{\alpha=0.75}$
ML	24/99	24/33	0.3636	0.2909
ML†	85/99	85/229	0.5183	0.6464
$\tilde{F}_{\alpha=0.5}$	85/99	85/211	0.5484	0.6693
$\tilde{F}_{\alpha=0.75}$	95/99	95/330	0.4429	0.6061

Table 3: Evaluation results

classifier was evaluated on the test dataset and its recall (R), precision (P), $F_{\alpha=0.5}$ measure, and $F_{\alpha=0.75}$ measure recorded. The results appear in Table 3.

The row labeled ML† is special: the classifier used here is the logistic regression model fitted by maximum likelihood; what is different is that the threshold for positive predictions was adjusted *post hoc* to match the number of true positives of the first discriminatively trained classifier. This has the same effect as manually adjusting the threshold parameter θ_0 based on partial knowledge of the test data (via the performance of another classifier) and is thus not permissible. It is interesting to note, however, that the ML trained classifier performs worse than the $\tilde{F}_{\alpha=0.5}$ trained classifier even when one parameter is adjusted by an oracle with knowledge of the test data and the performance of the other classifier.

Fitting a model based on $\tilde{F}_{\alpha=0.75}$, which gives increased weight to recall compared with $\tilde{F}_{\alpha=0.5}$, led to higher recall as expected. However, we also expected that the $F_{\alpha=0.75}$ score of the $\tilde{F}_{\alpha=0.75}$ trained classifier would be higher than the $F_{\alpha=0.75}$ score of the $\tilde{F}_{\alpha=0.5}$ trained classifier. This is not the case, and could be due to the optimization getting stuck in a local maximum, or it may have been an unreasonable expectation to begin with.

8 Conclusions

We have presented a novel estimation procedure for probabilistic classifiers which we call, by a slight abuse of terminology, *maximum expected F-measure* training. We made use of the fact that expected utility computations can be carried out in a vector space, and that an ordering of vectors can be imposed for purposes of maximization which can employ auxiliary functions like the F-measure (2). This technique is quite general and well suited for working with other quantities that can be expressed in terms of hits, misses, false alarms, correct rejections, etc. In particular, it could be used to find a point estimate which provides a certain tradeoff between specificity and sensitivity, or operating point. A more general method would try to optimize several such operating points simultaneously, an issue which we will leave for future research.

The classifiers discussed in this paper are logistic regression models. However, this choice is not crucial. The approximation (4) is reasonable for binary decisions in general, and one can use it in conjunction with any well-behaved conditional Bernoulli model or related classifier. For Support Vector Machines, approximate F-measure maximization was introduced by Musicant et al. (2003).

Maximizing F-measure during training seems especially well suited for dealing with skewed classes. This can happen by accident, because of the nature of the problem as in our summarization example above, or by design: for example, one can expect skewed binary classes as the result of the one-vs-all reduction of multi-class classification to binary classification; and in multi-stage classification one may want to alternate between classifiers with high recall and classifiers with high precision.

Finally, the ability to incorporate non-standard tradeoffs between precision and recall at training time is useful in many information extraction and retrieval applications. Human end-users often create asymmetries between precision and recall, for good reasons: they may prefer to err on the side of caution (e.g., it is less of a problem to let an unwanted spam email reach a user than it is to hold back a legitimate message), or they may be better at some tasks than others (e.g., search engine users are good at filtering out irrelevant documents returned by a query, but are not equipped to crawl the web in order to look for relevant information that was not retrieved). In the absence of methods that work well for a wide range of operating points, we need training procedures that can be made sensitive to rare cases depending on the particular demands of the application.

Acknowledgements

I would like to thank Julia Hirschberg, Sameer Maskey, and the three anonymous reviewers for helpful comments. I am especially grateful to

Sameer Maskey for allowing me to use his speech summarization dataset for the evaluation in Section 7. The usual disclaimers apply.

References

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

C. G. Broyden. 1967. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381.

D. R. Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society, Series B (Methodological)*, 20(2):215–242.

R. Fletcher. 1970. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322. doi:10.1093/comjnl/13.3.317.

R. Fletcher and C. M. Reeves. 1964. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154. doi:10.1093/comjnl/7.2.149.

Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. 2003. *GNU Scientific Library Reference Manual*. Network Theory, Bristol, UK, second edition.

Donald Goldfarb. 1970. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26.

Warren R. Greiff and Jay M. Ponte. 2000. The maximum entropy approach and probabilistic IR models. *ACM Transactions on Information Systems*, 18(3):246–287. doi:10.1145/352595.352597.

Abraham Ittycheriah, Lucian Lita, Nanda Kambhatla, Nicolas Nicolov, Salim Roukos, and Margo Stys. 2003. Identifying and tracking entity mentions in a maximum entropy framework. In *HLT/NAACL 2003*. ACL Anthology N03-2014.

Sameer Maskey and Julia Hirschberg. 2005. Comparing lexical, acoustic/prosodic, structural and discourse features for speech summarization. In *Interspeech 2005 (Eurospeech)*.

P. McCullagh and J. A. Nelder. 1989. *Generalized Linear Models*. Chapman & Hall/CRC, Boca Raton, FL, second edition.

David R. Musicant, Vipin Kumar, and Aysel Ozgur. 2003. Optimizing F-measure with Support Vector Machines. In *FLAIRS 16*, pages 356–360.

J. A. Nelder and R. W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society, Series A (General)*, 135(3):370–384.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL 41*. ACL Anthology P03-1021.

Chris Paciorek and Roni Rosenfeld. 2000. Minimum classification error training in exponential language models. In *NIST/DARPA Speech Transcription Workshop*.

Mazin Rahim and Chin-Hui Lee. 1997. String-based minimum verification error (SB-MVE) training for speech recognition. *Computer, Speech and Language*, 11(2):147–160.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Computer and Information Science.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *ARPA Human Language Technology Workshop*, pages 250–255. ACL Anthology H94-1048.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

D. F. Shanno. 1970. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656.