On Minimizing the Position Error in Label Ranking

Eyke Hüllermeier[1] and Johannes Fürnkranz[2]

[1] Department of Mathematics and Computer Science, Marburg University
eyke@mathematik.uni-marburg.de
[2] Department of Computer Science, TU Darmstadt
juffi@informatik.tu-darmstadt.de

Abstract. Conventional classification learning allows a classifier to make a one shot decision in order to identify the correct label. However, in many practical applications, the problem is not to give a single estimation, but to make repeated suggestions until the correct target label has been identified. Thus, the learner has to deliver a label ranking, that is, a ranking of all possible alternatives. In this paper, we discuss a loss function, called the position error, which is suitable for evaluating the performance of a label ranking algorithm in this setting. Moreover, we introduce "ranking through iterated choice", a general strategy for extending any multi-class classifier to this scenario, and propose an efficient implementation of this method by means of pairwise decomposition techniques.

1 Introduction

The main interest in the context of classification learning typically concerns the correctness of a prediction: A prediction is either correct or not and, correspondingly, is rewarded in the former and punished in the latter case. The arguably best-known loss function reflecting this problem conception is the misclassification or error rate of a classifier, that is, the probability of making an incorrect prediction. In this paper, we are interested in another scenario which motivates a generalization of the misclassification rate. As an illustration, consider a fault detection problem which consists of identifying the cause for the malfunctioning of a technical system. Suppose that a classifier has been trained to predict the true cause, e.g., on the basis of certain sensor measurements serving as input attributes (see, e.g., [1] for an application of that type). Now, if it turned out that a predicted cause is not correct, one cannot simply say that the classification process terminated with a failure. Instead, since the cause must eventually be found, alternative candidates must be tried until the problem is fixed.

What is needed in applications of this type is not only a prediction in the form of a single class label but instead a *ranking* of all candidate labels. In fact, a ranking suggests a simple (trial and error) search process, which successively tests the candidates, one by one, until the correct cause is found. An obvious measure of the quality of a predicted ranking is a loss function that counts the number of futile trials made before the target label is identified.

Apart from a suitable loss function, one needs a learner that produces label rankings as outputs. In this regard, the most obvious idea is to use a scoring classifier which outputs a score for each label, which is then used for sorting the labels. In particular, one may use a probabilistic classifier that estimates, for every candidate label λ, the conditional probability of λ given the input $\boldsymbol{x}$. Intuitively, *probabilistic ranking* (PR), i.e., ordering the labels according to their respective probabilities of being the target label, appears to be a reasonable approach.

In Section 3, we show that this approach is indeed optimal in a particular sense. Despite this theoretical optimality, however, an implementation of the approach turns out to be intricate in practice, mainly because estimating conditional probabilities is a difficult problem. In fact, it is well-known that most classification algorithms commonly used in the field of machine learning do not produce accurate probability estimates, even though they may have a strong hit rate. This motivates an alternative approach, to be introduced in Section 3.1, that we call *ranking through iterated choice* (RIC). The idea of this method is to employ a (multi-class) classifier as a choice function which, given a set of candidate labels and related training data, selects the most promising among these candidates. Roughly speaking, a label ranking is then obtained by repeated classification: In every iteration, the learning algorithm removes this label, and retrains a classifier for the remaining labels. Due to the retraining, RIC obviously comes along with an increased complexity. To overcome this problem, an efficient implementation of this approach, which is based on pairwise decomposition techniques, is proposed in Section 3.2. Experimental results, showing that RIC does indeed improve accuracy in comparison with PR, are presented in Section 4.

2 Label Ranking and Position Error

We consider a learning problem which involves an input space $\mathcal{X}$ and an output set $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$ consisting of a finite number of class labels. Assuming $\mathcal{X} \times \mathcal{L}$ to be endowed with a probability measure, one can associate a vector

$$p_{\boldsymbol{x}} = (\, \mathbb{P}(\lambda_1 \,|\, \boldsymbol{x}) \ldots \mathbb{P}(\lambda_m \,|\, \boldsymbol{x})\,) \tag{1}$$

of conditional class probabilities with every input $\boldsymbol{x} \in \mathcal{X}$, where $\mathbb{P}(\lambda_i \,|\, \boldsymbol{x}) = \mathbb{P}(\lambda_i = \lambda_{\boldsymbol{x}})$ denotes the probability that $\boldsymbol{x}$ belongs to class λ_i.

Given a set of training examples $\mathcal{D} = \{(\boldsymbol{x}_1, \lambda_{\boldsymbol{x}_1}) \ldots (\boldsymbol{x}_n, \lambda_{\boldsymbol{x}_n})\} \subset (\mathcal{X} \times \mathcal{L})^n$, the learning problem is to induce a "label ranker", which is a function that maps any input $\boldsymbol{x}$ to a total order of the class labels, i.e., a complete, transitive, and asymmetric relation $\succ_{\boldsymbol{x}}$ on $\mathcal{L}$; here, $\lambda_i \succ_{\boldsymbol{x}} \lambda_j$ means that λ_i precedes λ_j in the ranking associated with $\boldsymbol{x}$. Formally, a ranking $\succ_{\boldsymbol{x}}$ can be identified with a permutation $\tau_{\boldsymbol{x}}$ of $\{1 \ldots m\}$, e.g., the permutation $\tau_{\boldsymbol{x}}$ satisfying $\lambda_{\tau_{\boldsymbol{x}}^{-1}(1)} \succ_{\boldsymbol{x}}$

$\lambda_{\tau_{\boldsymbol{x}}^{-1}(2)} \succ_{\boldsymbol{x}} \ldots \succ_{\boldsymbol{x}} \lambda_{\tau_{\boldsymbol{x}}^{-1}(m)}$. Here, $\tau_{\boldsymbol{x}}(i) = \tau_{\boldsymbol{x}}(\lambda_i)$ is the position of label λ_i in the ranking.

In hitherto existing approaches to label ranking [4, 3], the quality of a prediction is measured in terms of a similarity or distance measure for rankings; for example, a commonly used measure for comparing a predicted ranking (permutation) $\tau_{\boldsymbol{x}}$ and a true ranking $\tau_{\boldsymbol{x}}^*$ is the Spearman rank correlation. Measures of that type take the position of *all* labels into account, which means, e.g., that swapping the positions of the two bottom labels is as bad as swapping the positions of the two top labels.

Measures such as Spearman rank correlation quantify, say, the *ranking error* of a prediction [5]. In this paper, we are interested in an alternative type of measure, which is especially motivated by practical performance tasks where a prediction is used in order to support the search for a true target label. As outlined in the introduction, an obvious loss function in this context is the number of labels preceding that label in the predicted ranking. Subsequently, a deviation of the predicted target label's position from the top-rank will be called a *position error*. Note that, while a ranking error relates to the comparison of two complete label rankings $\tau_{\boldsymbol{x}}$ and $\tau_{\boldsymbol{x}}^*$, the position error refers to the comparison of a label ranking $\tau_{\boldsymbol{x}}$ and a true class $\lambda_{\boldsymbol{x}}$. More specifically, we define the position error of a prediction $\tau_{\boldsymbol{x}}$ as $\mathrm{PE}(\tau_{\boldsymbol{x}}, \lambda_{\boldsymbol{x}}) \overset{\mathrm{df}}{=} \tau_{\boldsymbol{x}}(\lambda_{\boldsymbol{x}})$, i.e., by the position of the target label $\lambda_{\boldsymbol{x}}$ in the ranking $\tau_{\boldsymbol{x}}$. To compare the quality of rankings of different problems, it is useful to normalize the position error for the number of labels. This *normalized position error* is defined as

$$\mathrm{NPE}(\tau_{\boldsymbol{x}}, \lambda_{\boldsymbol{x}}) \overset{\mathrm{df}}{=} \frac{\tau_{\boldsymbol{x}}(\lambda_{\boldsymbol{x}}) - 1}{m-1} \in \{0, 1/(m-1) \ldots 1\}. \tag{2}$$

The position error of a label ranker is the *expected* position error of its predictions, where the expectation is taken with respect to the underlying probability measure on $\mathcal{X} \times \mathcal{L}$.

Compared with the conventional misclassification rate, the position error differentiates between "bad" predictions in a more subtle way: In the case of a correct classification, both measures coincide. In the case of a wrong top label, however, the misclassification rate is 1, while the position error assumes values between 1 and m, depending on how "far away" the true target label is.

Like most performance measures, the position error is a simple scalar index. To characterize a label ranking algorithm in a more elaborate way, an interesting alternative is to look at the mapping $C : \{1 \ldots m\} \rightarrow \mathbb{R}$ such that $C(k) = \mathbb{P}\left(\tau_{\boldsymbol{x}}(\lambda_{\boldsymbol{x}}) \leq k\right)$, i.e., $C(k)$ is the probability that the target label is among the top k labels in the predicted ranking. Of course, on the basis of this distribution, only a partial order can be defined on a class of learning algorithms: Two learners are incomparable in the case of intersecting C-distributions.

3 Minimizing the Position Error

What kind of ranking procedure should be used in order to minimize the risk of a predicted ranking with respect to the position error as a loss function? As mentioned before, an intuitively plausible idea is to order the candidate labels λ according to their probability $\mathbb{P}(\lambda = \lambda_{\boldsymbol{x}})$ of being the target label. In fact, this idea is not only plausible but also provably correct. Even though the result is quite obvious, we state it formally as a theorem.

Theorem 1. *Given a query instance $\boldsymbol{x} \in \mathcal{X}$, ranking the labels $\lambda \in \mathcal{L}$ according to their (conditional) probabilities of being the target class $\lambda_{\boldsymbol{x}}$ yields a risk minimizing prediction with respect to the position error (2) as a loss function. That is, the expected loss $\mathbb{E}(\tau_{\boldsymbol{x}}) = \frac{1}{m-1}\sum_{i=1}^{m}(i-1)\cdot \mathbb{P}\left(\tau_{\boldsymbol{x}}(\lambda_{\boldsymbol{x}}) = i\right)$ becomes minimal for any ranking $\succ_{\boldsymbol{x}}$ such that $\mathbb{P}(\lambda_i = \lambda_{\boldsymbol{x}}) > \mathbb{P}(\lambda_j = \lambda_{\boldsymbol{x}})$ implies $\lambda_i \succ_{\boldsymbol{x}} \lambda_j$.*

According to the above result, the top rank (first position) should be given to the label $\lambda_\top$ for which the estimated probability is maximal. Regarding the second rank, recall the fault detection metaphor, where the second hypothesis for the cause of the fault is only tested in case the first one turned out to be wrong. Thus, for the next choice, one has obtained additional information, namely that $\lambda_\top$ *is not the correct label.* Taking this information into account, the second rank should not simply be given to the label with the second highest probability according to the original probability measure, say, $\mathbb{P}_1(\cdot) = \mathbb{P}(\cdot)$, but instead to the label that maximizes the *conditional* probability $\mathbb{P}_2(\cdot) = \mathbb{P}(\cdot \,|\, \lambda_{\boldsymbol{x}} \neq \lambda_\top)$ of being the target label given that the first proposal was incorrect.

At first sight, passing from $\mathbb{P}_1(\cdot)$ to $\mathbb{P}_2(\cdot)$ may appear meaningless from a ranking point of view, since standard probabilistic conditioning yields

$$\mathbb{P}_2(\lambda) = \frac{1 - \mathbb{P}_1(\lambda)}{\mathbb{P}_1(\lambda_\top)} \propto \mathbb{P}_1(\lambda) \qquad (3)$$

for $\lambda \neq \lambda_\top$, and therefore does not change the order of the remaining labels. And indeed, in case the original $\mathbb{P}(\cdot)$ is a proper probability measure and conditioning is performed according to (3), the predicted ranking will not change at all.

3.1 Empirical Conditioning

One should realize, however, that standard conditioning is not an incontestable updating procedure in our context, simply because $\mathbb{P}_1(\cdot)$ is not a “true” probability measure over the class labels. Rather, it is only an estimated measure coming from a learning algorithm, perhaps one which is not a good probability estimator. In fact, it is well-known that most machine learning algorithms for classification perform rather poorly in probability estimation, even though they may produce good classifiers. Thus, it seems sensible to perform “conditioning” not on the measure itself, but rather on the learner that produced the measure. What we mean by this is that the learner should be retrained on the original data without the $\lambda_\top$-examples, an idea that could be paraphrased as “empirical conditioning”.

This type of conditioning depends on the data $\mathcal{D}$ and the model assumptions, that is, the hypothesis space $\mathcal{H}$ from which the classifier is taken. To emphasize this dependence and, moreover, to indicate that it concerns an *estimated* (“hat”)

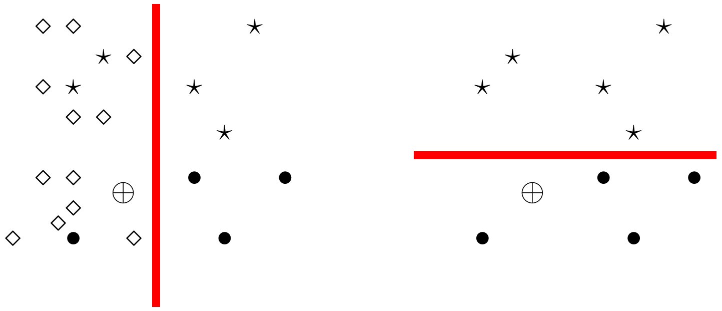

Fig. 1. Example of empirical conditioning: The optimal model (decision stump) for the complete training data (left) and the data omitting the examples of the top label ($\diamond$).

probability, the conditional measure $\mathbb{P}_2(\cdot)$ could be written more explicitly as $\mathbb{P}_2(\cdot) = \widehat{\mathbb{P}}(\cdot \,|\, \lambda_{\boldsymbol{x}} \neq \lambda_\top, \mathcal{D}, \mathcal{H})$. To motivate the idea of empirical conditioning, consider the simple example in Fig. 1, where the hypothesis space $\mathcal{H}$ is given by the class of decision stumps (univariate decision trees with only one inner node, i.e., axis-parallel splits in the case of numerical attributes). Given the examples from three classes (represented, respectively, by the symbols $\diamond$, $\star$, and $\bullet$), the best model corresponds to the split shown in the left picture. By estimating probabilities through relative frequencies in the leaf nodes of the decision stump, one derives the following estimates for the query instance, which is marked by a $\oplus$ symbol: $\widehat{\mathbb{P}}(\diamond \,|\, \oplus) = 12/15$, $\widehat{\mathbb{P}}(\star \,|\, \oplus) = 2/15$, $\widehat{\mathbb{P}}(\bullet \,|\, \oplus) = 1/15$; thus, the induced ranking is given by $\diamond \succ \star \succ \bullet$. Now, suppose that the top label $\diamond$ turned out to be an incorrect prediction. According to the above ranking (and probabilistic conditioning), the next label to be tested would be $\star$. However, when fitting a new model to the training data without the $\diamond$-examples, the preference between $\star$ and $\bullet$ is reversed, because the query instance is now located "on the $\bullet$-side" of the decision boundary. Roughly speaking, conditioning by "taking a different look" at the data, namely a look that suppresses the $\diamond$ examples, gives a quite different picture (shown on the right-hand side of Fig. 1) of the situation. In fact, one should realize that, in the first model, the preference between $\star$ and $\bullet$ is strongly biased by the $\diamond$-examples: The first decision boundary is optimal only because it classifies all $\diamond$-examples correctly, a property that looses importance once it turned out that $\diamond$ is not the true label of the query.

According to the above idea, a classifier is used as a *choice function*: Given a set of potential labels with corresponding training data (and a new query instance $\boldsymbol{x}$), it selects the most likely candidate among these labels. We refer to the process of successively selecting alternatives by estimating top-labels from (conditional) probability measures $\mathbb{P}_1(\cdot), \mathbb{P}_2(\cdot) \ldots \mathbb{P}_m(\cdot)$ as *ranking through iterated choice* (RIC). As an important advantage, note that this approach can be used to turn any multi-class classifier into a label ranker. In principle, it is not required that a corresponding classifier outputs a score, or even a real probability, for every label. In fact, since only a simple decision in favor of a single label has to be made in each iteration, any classifier is good enough. In this regard, let us note that, for the ease of exposition, the term "probability" will subsequently be used in a rather informal manner.

Regarding its effect on label ranking accuracy, one may expect the idea of RIC to produce two opposite effects: (1) *Information loss:* In each iteration, the size of the data set to learn from becomes smaller. (2) *Simplification:* Due to the reduced number of classes, the learning problems become simpler in each iteration. The first effect will clearly have a negative influence on generalization performance, as a reduction of data comes along with a loss of information. In contrast to this, the second effect will have a positive influence: The classifiers will become increasingly simple, because it can be expected that the decision boundary for separating m classes is more complex than the decision boundary for separating $m' < m$ classes of the same problem. The hope is that, in practice, the second (positive) effect will dominate the first one.

3.2 Efficient Implementation

An obvious disadvantage of RIC concerns its computational complexity. In fact, since empirical conditioning essentially means classifying on a subset of $\mathcal{L}$, the number of models needed is (potentially) of the order $2^{|\mathcal{L}|}$. To overcome this problem, we propose the use of pairwise decomposition techniques.

The idea of pairwise learning is well-known in the context of classification [2], where it allows one to transform a polychotomous classification problem, i.e., a problem involving $m > 2$ classes $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$, into a number of *binary* problems. To this end, a separate model (base learner) $\mathcal{M}_{ij}$ is trained for each *pair* of labels $(\lambda_i, \lambda_j) \in \mathcal{L}$, $1 \leq i < j \leq m$; thus, a total number of $m(m-1)/2$ models is needed. $\mathcal{M}_{ij}$ is intended to separate the objects with label λ_i from those having label λ_j. Depending on the classifier used, an output $\mathcal{M}_{ij}(\boldsymbol{x})$ can be interpreted, e.g., as the conditional probability $p_{ij} = \mathbb{P}\left(\lambda_{\boldsymbol{x}} = \lambda_i \,|\, \lambda_{\boldsymbol{x}} \in \{\lambda_i, \lambda_j\}, \boldsymbol{x}\right)$. In a second step, an estimation of the probability vector (1), i.e., of the individual probabilities $p_i = \mathbb{P}(\lambda_{\boldsymbol{x}} = \lambda_i \,|\, \boldsymbol{x})$, has to be derived from these pairwise probabilities. To this end, different techniques have been developed. Here, we resorted to the approach proposed in [7], which derives the p_i as a solution of a system of linear equations, S, that includes one equation for every label.

RIC can then be realized as follows: First, the aforementioned system of linear equations is solved, and the label λ_i with maximal probability p_i is chosen as the top-label $\lambda_\top$. This label is then removed, i.e., the corresponding variable p_i and its associated equation are deleted from S. To find the second best label, the same procedure is then applied to the reduced system S' thus obtained, i.e., by solving a system of $m-1$ linear equations and $m-1$ variables. This process is iterated until a full ranking has been constructed.

This approach reduces the training effort from an exponential to a quadratic number of models. Roughly speaking, a classifier on a subset $\mathcal{L}' \subseteq \mathcal{L}$ of classes is efficiently assembled "on the fly" from the corresponding subset of pairwise models $\{\mathcal{M}_{ij} \mid \lambda_i, \lambda_j \in \mathcal{L}'\}$. Or, stated differently, the *training* of classifiers is replaced by the *combination* of associated binary classifiers.

The hope that empirical conditioning improves accuracy in comparison with conventional probabilistic conditioning is essentially justified by the aforementioned *simplification effect* of RIC. Note that this simplification effect is also inherently present in pairwise learning. Here, the simplification due to a reduction of class labels is already achieved at the very beginning and, by decomposing the original problem into *binary* problems, carried to the extreme. Thus, if the simplification effect is indeed beneficial in the original version of RIC, it should also have a positive influence in the pairwise implementation (RIC-P). These are exactly the two conjectures to be investigated empirically in the next section: (i) Empirical conditioning (RIC) pays off with respect to accuracy, and (ii) the increased efficiency of the pairwise implementation, RIC-P, is achieved without sacrificing this gain in accuracy.

4 Empirical Results

In order to investigate the practical usefulness of empirical conditioning and the related RIC procedure, we compare the corresponding strategy to the most obvious alternative, namely ordering the class labels right away according to the respective probabilities produced by a multi-class classifier (probabilistic ranking, PR). So, given any multi-class classifier, capable of producing such probabilities, as a base learner, we consider the following three learning strategies: **PR:** A ranking is produced by applying the base learner to the complete data set only once and ordering the class labels according to their probabilities. **RIC:** This version refers to the *ranking through iterated choice* procedure outlined in Section 3.1, using the multi-class classifier as a base learner. **RIC-P:** This is the pairwise implementation of RIC as introduced in Section 3.2 (again using as base learners the same classifiers as RIC and PR). In cases of non-unique top-labels, we always break ties by coin flipping.

For 18 benchmark data sets from the UCI repository and the StatLib archive[3] we estimated the mean (absolute) position error of each method using leave-one-out cross validation, using two widely known machine learning algorithms as base learners: C4.5 and Ripper. For comparison purpose, we also derived results for the naive Bayes (NB) classifier, as this is one of the most commonly used "true" probabilistic classifiers. Note that, since conditional probabilities in NB are estimated individually for each class, empirical conditioning is essentially the same as conventional conditioning, i.e., RIC is equivalent to PR.

From the win-loss statistics for NB in comparison with PR using, respectively, C4.5 (10/8) and Ripper (10/8), there is no visible difference between these multi-class classifiers in terms of label ranking accuracy. Important are the win-loss statistics summarized in Table 1 (detailed results had to be omitted here due to space restrictions but can be found in an extended version of this paper [6]).

[3] `http://www.ics.uci.edu/~mlearn`, `http://stat.cmu.edu/`

Table 1. Win/loss statistics for each pair of methods, using C4.5 (left) and Ripper (right) as base learners.

	PR	RIC	RIC-P	PR	RIC	RIC-P
PR	—	3/13	4/13	—	3/13	3/12
RI	13/3	—	7/8	13/3	—	2/13
RIC-P	13/4	8/7	—	12/3	13/2	—

These results perfectly support the two conjectures raised above. First, RIC significantly outperforms PR: According to a simple sign test for the win-loss statistic, the results are significant at a level of 2%. Second, RIP-P is fully competitive to RIC (and actually shows a better performance in the case of Ripper as a base learner).

5 Concluding Remarks

In the context of the label ranking problem, we have discussed the position error as an alternative loss function. To minimize this loss function, we proposed *ranking through iterated choice* (RPC), a strategy that essentially reduces label ranking to repeated classification. In each iteration, RPC performs *empirical conditioning*, which in turn requires the retraining of classifiers. To avoid the need for training a potentially large number of models, we used a pairwise implementation in which retraining is done implicitly, namely by combining the outputs of certain pairwise models. In an experimental study, RPC was compared to standard probabilistic ranking, where the class labels are ranked according to the originally estimated probabilities. Our results suggest that retraining (empirical conditioning) does indeed reduce the expected loss when using standard multi-class classifiers as base learners, and that this gain in accuracy is preserved by the pairwise implementation.

References

1. C. Alonso, JJ. Rodríguez, and B. Pulido. Enhancing consistency based diagnosis with machine learning techniques. In *Current Topics in AI*, vol. 3040 of LNAI, 312–321. Springer, 2004.
2. J. Fürnkranz. Round robin classification. *J. of Mach. Learn. Res.*, 2:721–747, 2002.
3. J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Proc. ECML-03*, Cavtat-Dubrovnik, Croatia, 2003.
4. S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: a new approach to multiclass classification. In *Proc. ALT-02*, pp. 365–379, Lübeck, 2002.
5. E. Hüllermeier and J. Fürnkranz. Learning label preferences: Ranking error versus position error. In *Proc. IDA–2005*, Madrid, 2005.
6. E. Hüllermeier and J. Fürnkranz. On minimizing the position error in label ranking. Technical Report TUD-KE-2007-04, TU Darmstadt, 2007.
7. T.F. Wu, C.J. Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Machine Learning Res.*, 5:975–1005, 2004.