Efficient AUC Maximization with Regularized Least-Squares

Tapio PAHIKKALA [a], Antti AIROLA [a], Hanna SUOMINEN [a], Jorma BOBERG [a], and Tapio SALAKOSKI [a]

[a] *Turku Centre for Computer Science (TUCS), Department of Information Technology, University of Turku, Turku, Finland, firstname.lastname@utu.fi*

Abstract.
Area under the receiver operating characteristics curve (AUC) is a popular measure for evaluating the quality of binary classifiers, and intuitively, machine learning algorithms that maximize an approximation of AUC should have a good AUC performance when classifying new examples. However, designing such algorithms in the framework of kernel methods has proven to be challenging. In this paper, we address AUC maximization with the regularized least-squares (RLS) algorithm also known as the least-squares support vector machine. First, we introduce RLS-type binary classifier that maximizes an approximation of AUC and has a closed-form solution. Second, we show that this AUC-RLS algorithm is computationally as efficient as the standard RLS algorithm that maximizes an approximation of the accuracy. Third, we compare the performance of these two algorithms in the task of assigning topic labels for newswire articles in terms of AUC. Our algorithm outperforms the standard RLS in every classification experiment conducted. The performance gains are most substantial when the distribution of the class labels is unbalanced. In conclusion, modifying the RLS algorithm to maximize the approximation of AUC does not increase the computational complexity, and this alteration enhances the quality of the classifier.

1. Introduction

Classification problems constitute a typical supervised machine learning task domain, where the aim is to construct algorithms which predict for each input instance the class or classes to which it belongs. In binary classification, the task is to judge for every input instance whether it has a certain property (a positive example) or not (a negative example), and assign exactly one of two possible class labels accordingly. The task is often solved by mapping the input instances on a real-valued scale; the larger (smaller) the image is, the more confident the classifier is about the instance being a positive (negative) example. The binary output is then constructed by setting a threshold which divides the instances into positive and negative examples.

Evaluating the ability of the classifier to predict the class labels correctly is essential. Various criteria for this classification performance exist, and in performance evaluation, one must select a measure that reflects the chosen criterion. In binary classification, selecting the area under the receiver operating characteristics curve (AUC) measure has been recommended (see, e.g., [1,2,3]).

AUC corresponds to the probability that given a randomly chosen positive and negative example, the classifier will correctly distinguish them. Because AUC is calculated directly from the real-valued output, it has the potential to describe the classification performance in more detail than measures requiring a fixed threshold: if the performance of the classifier is measured by comparing only its binary output with the correct classification, the value of the performance evaluation measure may strongly depend on the threshold placement. Another advantage of AUC is its invariance to the distribution of class labels [4].

The desire for using AUC in performance evaluation has naturally led to the design of algorithms that aim to maximize AUC. In the framework of support vector machines, this task has proven to be challenging (see, e.g., [5,6,7]): The need to consider all positive-negative example pairs instead of individual examples easily leads to too expensive computations, and hence, approximative heuristics have had to be used to reduce the computational complexity. Moreover, the performance gains have often been modest, or not statistically significant.

In this paper, we address AUC maximization with the regularized least-squares (RLS) algorithm, also known as the least-squares support vector machine [8]. The standard RLS algorithm maximizes an approximation of the classification accuracy (ACC), that is, the proportion of correctly classified examples. It has been shown to achieve a classification performance similar to the regular support vector machines [9].

In our recent study [10], we introduced a RLS-based ranking algorithm that we call RankRLS, and applied it to the task of pairwise ranking of data points in information retrieval tasks. Although the number of possible data point pairs in such tasks grows quadratically with respect to the number of the individual data points, the computational complexity of the RankRLS algorithm was shown to be equal to the standard RLS regression. A similar algorithm was independently proposed by [11]. As the problem of AUC maximization can be naturally cast into the problem of comparing positive-negative data point pairs, RLS is a particularly suitable basis for developing an efficient AUC maximizing classifier.

Here, we introduce the AUC-RLS algorithm that maximizes regularized least-squares approximation of AUC for binary classification. It is based on the same approach as RankRLS. We show that AUC-RLS preserves the computational efficiency of the standard RLS algorithm, and outperforms RLS in maximizing AUC in testing.

2. Accuracy and Area Under ROC Curve

Let $\mathcal{X}$ be the input space that can be any set and let $\mathcal{Y} = \{1, -1\}$ be the output space. We call the set of possible input-output pairs $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ the example space. We say that $z = (x, y) \in \mathcal{Z}$ is a positive example if $y = 1$. Otherwise z is a negative example. Further, let us denote $\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$, and let $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$ be the hypothesis space. In supervised learning, we are given a number of training examples with known class labels that we use to select a hypothesis from $\mathcal{H}$ for prediction of the outputs of unseen examples. Formally, let $X = (x_1, \ldots, x_m) \in (\mathcal{X}^m)^{\mathrm{T}}$ to be a sequence of inputs, where $(\mathcal{X}^m)^{\mathrm{T}}$ denotes the set of row vectors of size m whose elements belong to $\mathcal{X}$. Further, we define $Y = (y_1, \ldots, y_m)^{\mathrm{T}} \in \mathcal{Y}^m$ to be a sequence of the corresponding output values. We also denote $z_i = (x_i, y_i)$, $1 \leq i \leq m$. Together, X and Y form a training set $S = (X, Y)$.

We now consider the performance measures that we use to evaluate how well the hypotheses perform in a prediction task. In the following measure definitions, we use the training data, but the measures can, of course, be analogously defined for any sequence of data points. The error rate indicates the proportion of correctly classified data points. Formally, the error rate of f measured with X and Y is

$$p_{\text{ERR}}(Y, f(X)) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |y_i - \text{sign}(f(x_i))|, \tag{1}$$

where $f(X) = (f(x_1), \ldots, f(x_m))^{\text{T}} \in \mathbb{R}^m$, and sign is the signum function. The constant $\frac{1}{m}$ is a normalizer ensuring that the result is always between 0 and 1. The ACC performance measure can be simply defined as $p_{\text{ACC}}(Y, f(X)) = 1 - p_{\text{ERR}}(Y, f(X))$.

The AUC measure, in turn, can be calculated from the following formula which is also called the Wilcoxon-Mann-Whitney statistic:

$$p_{\text{AUC}}(S, f(X)) = \frac{1}{m_+ m_-} \sum_{y_i=+1, y_j=-1} \frac{1}{2}(1 + \text{sign}(f(x_i) - f(x_j))),$$

where m_+ and m_- are the numbers of positive and negative examples, respectively (see [12] for a proof). Analogously, we also define a measure area over ROC curve (AOC) whose optimum is at 0 instead of 1:

$$\begin{aligned} p_{\text{AOC}}(S, f(X)) &= \frac{1}{m_+ m_-} \sum_{y_i=+1, y_j=-1} \frac{1}{2}(1 - \text{sign}(f(x_i) - f(x_j))) \\ &= \frac{1}{m_+ m_-} \sum_{y_i=+1, y_j=-1} \frac{1}{2}(\text{sign}(y_i - y_j) - \text{sign}(f(x_i) - f(x_j))), \end{aligned} \tag{2}$$

where we have also written the expression $\text{sign}(y_i - y_j)$ which is equal to 1, to emphasize that our aim is not to classify individual data points but pairs of them. Clearly, $p_{\text{AOC}}(S, f(X)) = 1 - p_{\text{AUC}}(S, f(X))$, and hence, when we aim to find a hypothesis that maximizes AUC, we can select the one minimizing AOC.

3. RLS and AUC-RLS Algorithms

Following [13], we consider algorithms for hypothesis selection in the framework of regularized kernel methods consisting of a cost function and a regularizer. The cost functions, that are usually approximations of the performance measures, indicate how large error a hypothesis has with respect to the training set. The purpose of the regularizer is to penalize too complex hypotheses that overfit at the training phase, and thus are not able to generalize to unseen data. In the framework, the hypothesis space $\mathcal{H}$ is so-called reproducing kernel Hilbert space determined by a positive definite kernel function k. Then, the learning algorithm that selects the hypothesis f from $\mathcal{H}$ is defined as

$$\mathcal{A}(S) = \operatorname*{argmin}_{f \in \mathcal{H}} J(f),$$

where

$$J(f) = c(f(X), Y) + \lambda \|f\|_k^2, \tag{3}$$

c is a real valued cost function, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_k$ is the norm in $\mathcal{H}$. By the generalized representer theorem [13], the minimizer of (3) has the following form:

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i), \tag{4}$$

where $a_i \in \mathbb{R}$ and k is the kernel function associated with the reproducing kernel Hilbert space mentioned above. For the training set, we define the symmetric $m \times m$ kernel matrix K to be a matrix whose elements are $K_{i,j} = k(x_i, x_j)$. For simplicity, we also assume that K is strictly positive definite. This can be ensured, for example, by performing a small diagonal shift. Using this notation, we rewrite $f(X) = KA$ and $\|f\|_k^2 = A^{\mathrm{T}} K A$ where $A = (a_1, \ldots, a_m)^{\mathrm{T}}$.

A natural way to minimize the training error would be to directly use the error rate (1) as a cost function. Similarly, when we aim to maximize AUC, the corresponding cost function to be minimized would be AOC (2). However, it is well-known that the use of this type of cost functions leads to intractable optimization problems. Therefore, instead of using (1) and (2), we use functions approximating them. For the error rate (1), we use the following type of least-squares approximation

$$c(Y, f(X)) = \sum_{i=1}^{m} (y_i - f(x_i))^2, \tag{5}$$

that is, the sum of least-squares errors made on each training example. Analogously for AOC (2), we use the following type of least-squares approximation

$$c(Y, f(X)) = \sum_{y_i=+1, y_j=-1} (y_i - y_j - f(x_i) + f(x_j))^2. \tag{6}$$

This cost function can be interpreted as least-squares error of regressing the label differences $y_i - y_j$ with the prediction differences $f(x_i) - f(x_j)$. We used a similar approach in [10], where a least-squares based cost function that compared all the the query-document pairs related to the same query was proposed for ranking tasks in document retrieval.

By substituting (5) into (3), we get the standard RLS algorithm whose solution can be obtained from

$$A = (K + \lambda I)^{-1} Y, \tag{7}$$

where A determines (4) [9]. The solution (7) can be obtained by inverting a $m \times m$ matrix with computational complexity $O(m^3)$. On the other hand, substituting (6) into (3) provides us an AUC maximization method we call AUC-RLS whose solution is

$$A = (LK + \lambda I)^{-1} L Y, \tag{8}$$

where L is a $m \times m$ matrix, whose entries are defined as

$$L_{i,j} = \begin{cases} m_- \text{ when } i = j \wedge y_i = 1 \\ m_+ \text{ when } i = j \wedge y_i = -1 \\ -1 \text{ when } y_i \neq y_j \\ 0 \quad \text{otherwise} \end{cases}, \tag{9}$$

where m_+ and m_- are the numbers of positive and negative examples, respectively.

The calculation of the solution (8) requires multiplications and inversions of $m \times m$ matrices. Both types of operations are usually performed with methods whose computational complexities are $O(m^3)$, and hence the complexity of AUC-RLS is equal to the complexity of the standard RLS.

4. Primal Form of AUC-RLS

For such cases where there are very large amounts of training data available, the cubic complexity of AUC-RLS training can be prohibitive. In the literature, the problem of finding a minimizer for (3) is known as the dual formulation. We next derive the primal form of the AUC-RLS algorithm that is applicable whenever the linear kernel is used. Similarly to the standard version of the RLS, the primal form is computationally more efficient than the dual form in cases where the dimensionality of the feature space is smaller than the number of training examples. This is formally shown below.

First, we assume a situation where the data points can be represented as real valued vectors whose dimension h corresponds to the dimensionality of the feature space, that is, $\mathcal{X} = \mathbb{R}^h$. Further, we assume that $h < m$. Now, the sequence of inputs can be written as a matrix $X \in \mathbb{R}^{h \times m}$. If we consider only the linear kernel, the function (4) minimizing (3) can be equivalently expressed as

$$f(x) = x^{\mathrm{T}} X A = x^{\mathrm{T}} w, \tag{10}$$

where $w = XA$ denotes the normal vector of the hyperplane that determines the RLS solution. The vector, as we have shown in [10], can be obtained from

$$w = (XLX^{\mathrm{T}} + \lambda I)^{-1} XLY. \tag{11}$$

Calculating (11) involves matrix multiplications of $O(hm^2)$ complexity. However, we can speed up the calculation in the following way.

Without losing generality, we can re-index the training examples so that the indices $1, \ldots, m_+$ are assigned to the positive and the indices $m_+ + 1, \ldots, m$ to the negative examples. Now the corresponding L can be decomposed into $L = D - PQ$, where D is a diagonal matrix whose diagonal elements are given as $D_{i,i} = L_{i,i}$ and P and Q are defined as

$$P = \begin{pmatrix} \mathbf{1}_{m_+ \times 1} & \mathbf{0}_{m_+ \times 1} \\ \mathbf{0}_{m_- \times 1} & \mathbf{1}_{m_- \times 1} \end{pmatrix}, Q = \begin{pmatrix} \mathbf{0}_{1 \times m_+} & \mathbf{1}_{1 \times m_-} \\ \mathbf{1}_{1 \times m_+} & \mathbf{0}_{1 \times m_-} \end{pmatrix}.$$

Now (11) can be re-written as

$$w = ((XD)X^{\mathrm{T}} - (XP)(QX^{\mathrm{T}}) + \lambda I)^{-1}(X(DY) - X(P(QY))). \qquad (12)$$

The computational complexity of the matrix inversion is $O(h^3)$. The multiplication $(XD)X^{\mathrm{T}}$ can be performed in $O(h^2m)$ time. All other matrix operations in (12) have lower computational complexity. Thus the resulting overall complexity of primal AUC-RLS is $O(h^3 + h^2m)$, making the method preferable to the dual version whenever m is considerably larger than h.

5. Experiments

We evaluated the capability of the AUC-RLS algorithm to maximize AUC on a real world dataset by considering a well-known text classification problem: the assignment

Table 1. Comparison of the performance of the AUC-RLS and RLS algorithms in terms of AUC on the Reuters-21578 dataset. In the first column is the name of the predicted class and in the next two are the AUC-values for the tested algorithms with 95% confidence intervals in parentheses. The last two present the numbers of positive examples in the training set of 500 documents and test set of 12397 documents.

class	AUC-RLS	RLS	pos. train set	pos. test set
acq	0.980 (0.978–0.983)	0.979 (0.977–0.982)	94	2275
bop	0.966 (0.947–0.985)	0.880 (0.843–0.917)	4	101
cocoa	0.931 (0.891–0.970)	0.837 (0.776–0.899)	2	71
coffee	0.969 (0.948–0.990)	0.962 (0.950–0.975)	5	134
corn	0.970 (0.959–0.982)	0.950 (0.936–0.964)	11	226
cpi	0.947 (0.925–0.969)	0.601 (0.555–0.648)	3	94
crude	0.976 (0.969–0.982)	0.975 (0.969–0.982)	23	555
dlr	0.971 (0.961–0.981)	0.946 (0.926–0.965)	10	165
earn	0.994 (0.993–0.995)	0.993 (0.991–0.994)	158	3806
gnp	0.987 (0.981–0.993)	0.923 (0.891–0.956)	5	131
gold	0.970 (0.953–0.986)	0.922 (0.897–0.948)	4	120
grain	0.979 (0.973–0.985)	0.974 (0.968–0.980)	23	559
interest	0.965 (0.956–0.974)	0.952 (0.941–0.962)	19	459
livestock	0.701 (0.642–0.761)	0.637 (0.578–0.696)	3	96
money-fx	0.954 (0.946–0.962)	0.947 (0.938–0.957)	28	689
money-supply	0.949 (0.930–0.968)	0.907 (0.877–0.937)	7	165
nat-gas	0.957 (0.933–0.981)	0.941 (0.920–0.962)	5	100
oilseed	0.898 (0.877–0.919)	0.816 (0.783–0.849)	6	165
reserves	0.943 (0.908–0.977)	0.511 (0.458–0.564)	2	71
ship	0.949 (0.934–0.963)	0.925 (0.907–0.942)	13	273
soybean	0.876 (0.839–0.913)	0.805 (0.757–0.853)	4	107
sugar	0.985 (0.979–0.991)	0.964 (0.952–0.976)	6	156
trade	0.978 (0.970–0.986)	0.969 (0.960–0.977)	20	466
veg-oil	0.890 (0.865–0.914)	0.697 (0.656–0.739)	4	120
wheat	0.984 (0.978–0.990)	0.976 (0.969–0.983)	12	271

of topic labels for Reuters newswire articles. Our approach was to transform the problem into a series of binary classification tasks, and to compare the AUC performance of the AUC-RLS and standard RLS algorithms on each of these sub-tasks.

Our experiments were conducted on the Reuters-21578 dataset[1]. To simulate a situation where only a very limited amount of data is available, we extracted a representative set of 500 documents for training purposes. The extraction was performed so that the class distributions in the extracted subset were guaranteed to be approximately the same as in the whole dataset. The rest of the documents were reserved for final validation.

We considered the task of predicting the 25 most numerous classes, each separately using one-vs-all approach. All of these classes had at least two positive examples in the training data. Some of the documents belonged to more than one of the considered classes and some to none of them. Thus it was possible for a document to be a positive example in more than one of the 25 classification tasks, or in none of them.

In the tests we applied the linear kernel. Ten-fold cross-validation was used on the training data to choose the values of the regularization parameter λ individually for each class. In each case the parameter that produced maximal AUC taken over all of the folds was chosen. Fold partitions were stratified separately for each classification task. The classifiers were trained on the 500 training documents using the chosen parameters, and then tested on the 12397 test documents. To evaluate the statistical significance of the results on the level of individual classes we calculated the 95% confidence intervals for the classifiers' AUC scores for each class. These intervals were obtained with SPSS 11.0.

The results are summarized in Table 1. All the AUC-values are, by definition, real numbers between 0 and 1, 0.5 being the random baseline. AUC-RLS outperformed the standard RLS on each of the 25 classification tasks. When considered together, the results clearly show the difference between the performances of AUC-RLS and RLS to be statistically significant. It should be noted that for some classes the difference is notably larger than for some others. The greatest performance differences can be found in such cases where the class distributions are most unbalanced (e.g., cocoa, cpi, reserves). For the largest classes (e.g., acq, earn, crude) these differences are much more modest, or even negligible. To summarize, AUC-RLS performed reliably, whereas the standard RLS gave in many cases much worse results.

6. Conclusion

The main outcome of this study is a computationally efficient algorithm, AUC-RLS, that outperformed the standard RLS algorithm in maximizing AUC. The performance gains achieved by using the AUC-RLS algorithm were emphasized when the distribution of the class labels was strongly unbalanced. In conclusion, AUC-RLS retains the computational efficiency of the standard RLS algorithm and clearly improves AUC performance.

Our experiments consider AUC maximization with RLS in the binary newswire article classification task. As usual, the generalizability of the results to other application domains and machine learning tasks is limited. However, the achieved performance gains encourage us to study the performance of AUC-RLS in other domains. In particular, we anticipate AUC-RLS to be an attractive method for many real-world classification problems because the class label distribution is typically strongly unbalanced in reality. In

[1] Available at http://www.daviddlewis.com/resources/testcollections/reuters21578

the future, we plan to design efficient cross-validation algorithms for AUC-RLS in ways similar to the ones described by us in [14,15].

Acknowledgments

This work has been supported by the Academy of Finland and Tekes, the Finnish Funding Agency for Technology and Innovation.

References

[1] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[2] Foster J. Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann Publishers Inc., 1998.

[3] Jin Huang and Charles X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.

[4] Tom Fawcett and Peter A. Flach. A response to Webb and Ting's on the application of ROC analysis to predict classification performance under varying class distributions. *Machine Learning*, 58(1):33–38, 2005.

[5] Alain Rakotomamonjy. Optimizing area under ROC curve with SVMs. In José Hernández-Orallo, César Ferri, Nicolas Lachiche, and Peter A. Flach, editors, *Proceedings of the 1st International Workshop on ROC Analysis in Artificial Intelligence*, pages 71–80, 2004.

[6] Ulf Brefeld and Tobias Scheffer. AUC maximizing support vector learning. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.

[7] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 377–384. ACM Press, 2005.

[8] Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[9] Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

[10] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jorma Boberg, and Tapio Salakoski. Learning to rank with pairwise regularized least-squares. In Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai, editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

[11] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-preserving ranking algorithms. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning*, pages 169–176. Omnipress, 2007.

[12] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[13] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In D. Helmbold and R. Williamson, editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.

[14] Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Fast n-fold cross-validation for regularized least-squares. In Timo Honkela, Tapani Raiko, Jukka Kortela, and Harri Valpola, editors, *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence*, pages 83–90. Otamedia Oy, 2006.

[15] Tapio Pahikkala, Hanna Suominen, Jorma Boberg, and Tapio Salakoski. Transductive ranking via pairwise regularized least-squares. In Paolo Frasconi, Kristian Kersting, and Koji Tsuda, editors, *Workshop on Mining and Learning with Graphs*, pages 175–178, 2007.