21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Software-based Performance and Complexity Analysis for the Design of Embedded Classification Systems

Matthias Ring, Ulf Jensen, Patrick Kugler and Bjoern Eskofier
Digital Sports Group, Pattern Recognition Lab, University of Erlangen-Nuremberg, Germany
matthias.ring@informatik.stud.uni-erlangen.de, {ulf.jensen, patrick.kugler, bjoern.eskofier}@cs.fau.de

Abstract

Embedded microcontrollers are employed in an increasing number of applications as a target for the implementation of classification systems. This is true for example for the fields of sports, automotive and medical engineering. However, important challenges arise when implementing classification systems on embedded microcontrollers, which is mainly due to limited hardware resources.

In this paper, we present a solution to the two main challenges, namely obtaining a classification system with low computational complexity and at the same time high classification accuracy. For the first challenge, we propose complexity measures on the mathematical operation and parameter level, because the abstraction level of the commonly used Landau notation is too high in the context of embedded system implementation. For the second challenge, we present a software toolbox that trains different classification systems, compares their classification rate, and finally analyzes the complexity of the trained system. To give an impression of the importance of such complexity measures when dealing with limited hardware resources, we present the example analysis of the popular Pima Indians Diabetes data set, where considerable complexity differences between classification systems were revealed.

1. Introduction

There has been an increase in studies that implement classification systems on embedded devices, which is due to the miniaturization and increasing performance of microcontrollers and sensors. Examples can be found in the brain-computer interface field [9], the digital sports field [2], or the computer vision field [7].

All these studies faced challenges that emerge when designing perfect-fitting hardware-software combinations. We identified the major challenges and grouped them into two categories. The first category are applications where classification systems with preferably high classification accuracy are designed. The challenge is to find portable hardware for efficient execution, but for reasons of economy no hardware resources should be unused. The second category are applications where the hardware selection is constrained by the price, size and energy consumption of the hardware components. The challenge is to find the best-performing classification system within these constraints.

To handle the challenges in both categories, complexity knowledge about classification systems is necessary. This is commonly expressed in the Landau notation [10] that gives asymptotical space and time bounds. For pattern recognition algorithms, the Landau complexity can be found in standard literature as [1]. However, for the usage in the context of embedded systems the abstraction level of this notation is too high. First, explicit algorithm runtimes cannot be determined, only asymptotical bounds are given. Second, different operations with different runtimes are treated equally. Third, multiple operations can be combined and interpreted as one operation. To express more precise information, complexity measures for the usage in embedded systems should be based on the environment where the algorithms are executed. This environment is an embedded microcontroller that mostly executes mathematical instructions and processes data in numerical formats. Hence, we propose runtime measures on the level of mathematical operations and memory measures on the level of mathematical parameters. To date we are unaware of any study that used this approach.

In this paper, we present our analysis method, and software that trains and analyzes classification systems. We also present the example analysis of the Pima Indians Diabetes data set and demonstrate the selection of an appropriate classification system for an embedded implementation of this data set.

2. Theoretical analysis

In the training phase of classification systems, preprocessing, feature selection, classification and evaluation algorithms are performed. We did not consider feature extraction algorithms, i.e. we expected the input consisting of features, not raw sensor data.

978-4-9906441-1-6

However, interesting for the implementation on embedded devices is the working phase where only preprocessing and classification algorithms are executed. Hence, these algorithms determine the complexity of classification systems and needed to be analyzed. This analysis can be divided into two steps. The first step, the theoretical algorithm analysis, contains variables that cannot be resolved until the classification system is trained. An example is the number of features which is different in each data set and also may be reduced in the feature selection step. To generate a variable-free analysis result, we implemented a software that performs the second step: it trains classification systems and extracts the necessary information to resolve the variables from the first, theoretical analysis.

2.1. Performance and complexity measures

We defined three groups of measures to express the performance and complexity of a classification system: (1) the classification accuracy of the classifier, (2) the memory requirements for parameters of the classification system, and (3) the number and types of operations for one classification decision. We considered arithmetic operations, comparisons and basic mathematical function evaluations as operations. Additions and subtractions were handled as the same operation. The memory requirements are determined by the number and types of algorithm parameters that have to be saved on the embedded system. We distinguished between integers (INT) and floating point numbers (FLOAT).

2.2. Analysis of preprocessing algorithms

We considered two preprocessing algorithms in the analysis. The first algorithm is feature-wise normalization (NORM, [14]) that maps all numeric features into the same range. This levels the influence of different features with different value ranges. The second algorithm is outlier detection based on quartiles [8].

As an example, the analysis of the NORM algorithm is shown, the analysis for the second algorithm works similar. The d-th feature of an input pattern $\hat{\mathbf{x}}$ is denoted by $\hat{x}_d$. In the training phase, the algorithm determines the minimum value d_{min} and maximum value d_{max} for each feature. The user defines the scale factor s and translation constant t . In the working phase, the normalized pattern $\mathbf{x}$ is computed feature-wise with

$$x_d = s\frac{\hat{x}_d - d_{min}}{d_{max} - d_{min}} + t \ . \qquad (1)$$

For the analysis, we counted the number and types of operations and parameters that are necessary in Eq. (1) to normalize one input pattern. Tab. 1 summarizes the analysis of one normalization.

Table 1. The analysis of one normalization. The first three columns contain the number and types of operations, the last column contains the FLOAT parameters. The resulting numeric values depend on the number of features N_F.

$+,-$	$\cdot$	$\div$	FLOAT
$3N_F$	N_F	N_F	$2+2N_F$

2.3. Analysis of classification algorithms

The No Free Lunch theorem [1] states that there is no best classifier for all classification tasks. Comparing different classifiers is a mandatory step and therefore we analyzed several classification algorithms: linear discriminant analysis (LDA, [1]), support vector machine (SVM, [15]), naive Bayes (NB, [1]), nearest neighbor (NN, [1]), C4.5 [11], multi-layer perceptron (MLP, [1]), PART [4] and AdaBoost.M1 (AB, [1]).

Most algorithms have a fixed number of operations for one classification so that we can calculate the exact number. However, some algorithms have a varying number: C4.5, NN and PART. For example, the C4.5 algorithm builds a decision tree that may not be balanced and hence has a varying number of decisions for different input patterns. In such situations, we assumed the worst case in the theoretical analysis.

As an example, the analysis of the SVM with a radial basis kernel is shown. The analysis for the other algorithms works similar. In brief, the SVM transforms input patterns into a higher dimension and then finds the affine decision boundary that generates the maximum margin of separation between the transformed patterns.

$$\mathrm{f}(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{N_{SV}} y_i \alpha_i e^{-\gamma\|\mathbf{x_i}-\mathbf{x}\|_2^2} - b\right) \qquad (2)$$

In Eq. (2), the function f is the decision boundary, $\mathbf{x}$ the input pattern and N_{SV} the number of support vectors. For each support vector $\mathbf{x_i}$ the variable y_i denotes its class label and α_i its Lagrange multiplier. The variable b is a constant and γ the radial basis kernel parameter. For the analysis, we counted the number and types of operations and parameters that are necessary in Eq. (2) to classify one input pattern. Tab. 2 summarizes the analysis of one classification decision.

Table 2. The analysis of one SVM classification. The first four columns contain the number and types of operations, the next two columns contain the FLOAT and INT parameters. The resulting numeric values depend on the number of support vectors N_{SV} and the number of features N_F.

$+,-$	$\cdot$	e^x	$\leq$	FLOAT	INT
$2N_FN_{SV}$	$(3+N_F)\cdot N_{SV}$	N_{SV}	1	$(1+N_F)\cdot N_{SV}+2$	N_{SV}

3. Software-based analysis

Our software uses existing software packages like WEKA [5] to train the classification systems. It includes the mentioned preprocessing and classification algorithms. Additionally, it provides feature selection methods. The feature selection algorithm itself has no impact on the analysis, because it is not performed in the working phase. However, feature selection reduces the number of features and can achieve a better generalization of the classifier [13]. That in turn influences the accuracy and some variables in the analysis (e.g. N_F). We considered two methods: an exhaustive search [13] and a best first search (BF, [13]). The BF starts with an empty feature set (full feature set) and sequentially adds (removes) the feature that improves the rating criterion the most. For both methods, we computed the rating criterion with the wrapper method [6] that trains the classifier with the current feature set and evaluates the accuracy with a cross validation.

We included three evaluation methods to calculate the classification accuracy. First, the n-fold cross-validation (CV, [1]) that divides the training data into n subsets, and uses each subset once for testing and the remaining subsets for training. Second, the bootstrapping method [1] that randomly selects patterns with replacement, trains the classifier with this set and tests with the remaining, non-selected training patterns. Third, the training-test-set method [13] that uses two different data sets for training and testing.

The Embedded Classification Software Toolbox (ECST, see Fig. 1) that trains and analyzes classification systems with the presented method can be downloaded from our website: www.tinyurl.com/ecst-project.

4. Example analysis

We present an example analysis for the Pima Indians Diabetes data set [3]. In the discussion we demonstrate how to select an appropriate classification system for an embedded implementation of this data set based on the analysis results.

The data set is a two class problem with 768 training patterns from diabetes patients and healthy controls. Each one has eight numeric features and a class label. We executed experiments with all analyzed classifiers. The preprocessing, feature selection and evaluation methods were kept constant: NORM, BF and 10-fold CV, respectively. For any configurable options in the algorithms, the default values from WEKA were used. For the SVM a radial basis kernel (γ = 0.5) was used and different values for the C-parameter (0.5, 1, 2, 4 and 8) were tested.

5. Results

In the SVM system the best C-parameter value was 0.5, the BF method selected N_F = 6 features and

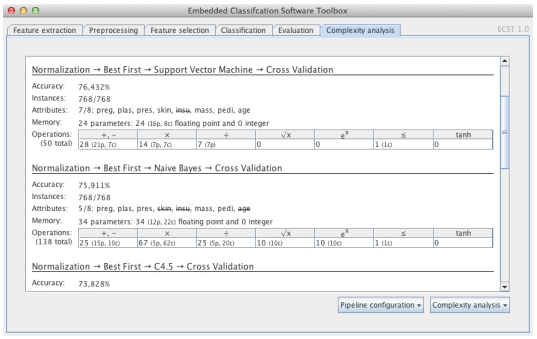

Figure 1. A screenshot of the ECST with the analysis result.

the SVM used N_{SV} = 473 support vectors. Our software combined these training results with the theoretical analysis from Tab. 1 and 2. The result is shown in Tab. 3. The SVM system achieved the highest accuracy (77.73%), followed by the LDA system (76.43%) and the NB system (75.91%). Although the SVM achieved the best accuracy, it also used the most operations and memory. The C4.5 classification system used the fewest operations (27 operations total) and required considerably fewer memory compared to the SVM system. The LDA system required the fewest memory (28 parameters total) and used considerably fewer operations compared to the SVM system. Besides NN, the accuracy of the different systems varied in a small range, but all achieved similar results as in the original publication of this data set [12].

6. Discussion

The classification system with the highest accuracy (SVM) also required the most memory and operations. However, other systems provided comparable accuracy with considerably fewer resources. To reduce the number of operations performed on the embedded system, the C4.5 system is the first choice. With 99.7% fewer operations and 97.5% fewer memory compared to the SVM system, it achieved only 4% less accuracy. Recall that the C4.5 analysis is a worst case scenario and the average operations are even fewer. To reduce the memory requirements on the embedded system, the LDA system is the first choice. With 99.2% fewer memory and 99.5% fewer operations compared to the SVM system, it achieved comparable accuracy. Additionally, both systems avoid the computationally expensive exponential function evaluations in the SVM kernel.

Some microcontrollers have limited instruction sets, for example no hardware implementation of multiplication and division. These operations would have to be implemented in software, which usually consumes more runtime than a hardware implementation. On such embedded systems, the C4.5 classifier is the first choice, because only comparisons are used. The few additions,

Table 3. Analysis of the Pima Indians Diabetes data set. The first column names the classifier, the second column contains the accuracy (ACC), the next three columns contain the number of FLOAT and INT parameters, and the number and types of operations for one classification decision follow. The table is sorted by the total number of operations in the last column.

Classifier	ACC (%)	FLOAT	INT	Total	$+, -$	$\cdot$	$\div$	$\sqrt{x}$	e^x	$\leq$	Total
SVM	77.73	3327	473	3800	5695	4263	6	0	473	2	10439
NN	65.62	772	769	1541	4	769	1	768	0	768	2310
NB	75.91	34	0	34	25	65	25	10	10	1	136
MLP	75.78	38	0	38	41	26	10	0	5	1	83
LDA	76.43	28	0	28	30	18	6	0	0	1	55
PART	73.44	31	42	73	18	6	6	0	0	17	47
AB	73.31	30	30	60	22	4	4	0	0	11	41
C4.5	73.83	20	73	93	12	4	4	0	0	7	27

multiplications and divisions are due to the NORM preprocessing.

Hence, our software creates a comprehensive overview of performance, memory requirements and computational costs of classification systems specific for each training data set. Using the analysis results in the development of embedded classification systems can reduce development time, production costs and time to market. However, there are drawbacks that have to be approached in further studies. We did not weight operations according to their runtime. For example, exponential function evaluations should be weighted higher than additions or subtractions. This is the same case with the memory measures (INT and FLOAT) that can consume different amounts of memory on a microcontroller.

Even though there is future work, the presented method is applicable in the general development of embedded systems, not only in the pattern recognition field. It is an inspiration to unify complexity considerations when working with limited hardware resources.

7. Outlook

In future versions, we are planning to integrate a feature selection method for the specific case of embedded systems. Based on user-chosen weights for each feature, the search will be regulated to select discriminant features that have less computational costs. The weights will be assigned to represent differences in the computational costs. An upper weight bound could restrict the overall computational cost of the final feature set.

8. Acknowledgements

Financial support was provided by the Embedded Systems Institute (ESI) Erlangen, supported in part by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology and the European Fund for Regional Development.

References

[1] R. O. Duda et al. *Pattern classification*. Wiley, 2001.

[2] B. Eskofier et al. Embedded surface classification in digital sports. *Pattern Recogn. Lett.*, 30(16):1448–1456, 2009.

[3] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, visited: 02/22/2012.

[4] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc. of the 15th ICML*, pages 144–151, Madison, USA, 1998.

[5] M. Hall et al. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[6] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[7] R. Kumaraswamy et al. SVM based classification of traffic signs for realtime embedded platform. *CCIS*, 193(3):339–348, 2011.

[8] J. Laurikkala et al. Informal identification of outliers in medical data. In *Proc. of the 5th IDAMAP*, pages 20–24, Berlin, Germany, 2000.

[9] C. Lin et al. Development of wireless brain computer interface with embedded multitask scheduling and its application on real-time driver's drowsiness detection and warning. *IEEE T. on Bio.-Med. Eng.*, 55(5):1582–1591, 2008.

[10] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[11] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[12] J. Smith et al. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proc. of the SCAMC*, pages 261–265, Washington, D.C., USA, 1988.

[13] S. Theodoridis and K. Koutroumbas. *Pattern recognition*. Academic Press, 2006.

[14] S. Tufféry. *Data Mining and Statistics for Decision Making*. Wiley, 2011.

[15] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.