10.24425/acs.2025.157142

Archives of Control Sciences
Volume 35(LXXI), 2025
No. 4, *pages* 655–681

Construction of multiclass classifier as linear or mixed binary programming task

Katerina HORAISOVA, Jaromir KUKAL and Pavel CEJNAR

Linear and mixed binary programming techniques, which arise from model linearity, are widely supported by advanced optimization solvers. In this paper, we present a comprehensive guide for transforming linear classifiers into linear or mixed binary programming tasks. Our approach employs widely used techniques, such as Kesler construction with either perfect or imperfect learning, and weight regularization using the L_1 or L_0 norm, enhanced by additional maximum weight constraint (i.e., L_∞). Various linear optimization tasks are formulated based on performance measures such as accuracy and sensitivity.

The classifiers are constructed using different weight penalizations and regularizations – specifically, the L_0 norm, which yields mixed binary programming tasks with NP-hard complexity, and the L_1 norm, which results in linear programming tasks with polynomial complexity, both with an additional maximum weight constraint. The proposed classifiers are compared on several UCI datasets (Iris Flower, Wine, and Seeds) and match or outperform Ridge and Lasso regression methods when applied to classification tasks.

Key words: linear classifier, Kesler construction, weight number reduction, feature selection, linear programming, mixed binary programming

1. Introduction

Constructing multiclass classifiers [19, 27, 31] is a traditional task in pattern recognition, closely linked to modern machine learning techniques. Many pattern recognition methods arise from linear and nonlinear iterative schemes, stochastic

K. Horaisova (corresponding author, e-mail: katerina.horaisova@fjfi.cvut.cz) is with FNSPE CTU in Prague, Trojanova 13, Prague, 120 00, Czech Republic.

J. Kukal (e-mail: jaromir.kukal@vscht.cz) and P. Cejnar (e-mail: pavel.cejnar@vscht.cz) are with FCE UCT in Prague, Technicka 5, Prague 6, 166 28, Czech Republic.

The paper is supported by the SGS23/190/OHK4/3T/14 grant of the Czech Technical University in Prague. P.C. acknowledges support for algorithmic and statistical software optimization by the Ministry of Education, Youth and Sports of the Czech Republic, grant INTER-COST LUC23138.

Received 26.06.2025. Revised 13.10.2025.

optimization processes, linear statistical models, quadratic programming, and other advanced approaches. When a fixed nonlinear transformation (a kernel function) is allowed to apply on the input patterns, the resulting classifier can be even linear. Regularized linear models – such as Lasso [44] and Ridge [21,22,40] regression – applied in the feature space induced by a kernel transformation have been shown to handle nonlinear classification tasks effectively; similarly, support vector machines (SVMs [43]) exploit this principle. By mapping the input data into a higher-dimensional space via a fixed nonlinear transformation, these methods combine the expressive power of kernel methods with the simplicity and computational efficiency of linear classifiers. Another example of this approach is the RVFL network [33], which employs Ridge regression after a randomly designed nonlinear transformation of the patterns.

Over the past five years, progress in general-purpose multiclass classification has centered on ever-richer regularization schemes for linear or kernelized models, such as the elastic net Extreme Learning Machine (ELM) [18], the Liu-ELM [51] and its variants such as Liu-Lasso ELM [52] or a combination of Ridge and Liu regressions for ELM [53], and the conjugate gradient optimized regularized ELM [26], all of which consistently surpass classic Ridge or Lasso baselines. Complementary ideas impose structural constraints on the label space, exemplified by the denoising low-rank discriminative least-squares regressor [25], while instance- and prototype-oriented sparsity has been explored through the group-Lasso sparse-KNN [56] and the Ranking-based Instance Selection pre-precessor [8]. Ensemble innovations such as the two-stage Duet classifier [46] further balance accuracy and computational cost, and even deep networks have adopted more expressive penalties, e.g. the soft-diamond regularizer [1], to outperform standard L1/L2 constraints. Despite these advances, almost none of the recent methods reformulate multiclass learning as a linear or mixed-integer programming problem, leaving the rich toolbox of exact optimization techniques largely untapped. Addressing this gap, our work casts multiclass classification directly as a linear or mixed-binary program, providing a transparent, solver-ready alternative to the prevailing heuristic regularization approaches.

Modern optimization solvers for linear programming (LP) [13, 42, 45] and mixed binary programming (MBP) [34] tasks have scaled to handle large problems, opening new research areas such as best subset selection for large datasets [5] or the analysis of neural network features via adversarial examples [29]. Our study demonstrates the practical feasibility of constructing a multiclass classifier by transforming advanced classification schemes – such as the Kesler construction – combined with regularization techniques (e.g., Ridge regression or a similar one) into equivalent LP or MBP tasks. These tasks can then be solved using external optimization solvers. For linearly nonseparable datasets, a linear clas-

sifier inevitably incurs some error because it cannot perfectly separate the data. This limitation is addressed either by applying a fixed nonlinear transformation (the kernel trick) before using a linear classifier or by introducing compensatory variables for imperfect learning.

For multiclass classification, the Kesler construction [16, 20] directly generates the target class index for each pattern, rather than relying on a one-vs-all scheme (i.e., selecting the highest output value). Although implementing this method directly requires advanced optimization techniques, reformulating it as an equivalent LP or MBP task is well suited for use with external solvers. Whether insisting on perfect classification of the target class index or allowing some misclassifications in exchange for improved robustness, both approaches can be effectively handled by the linear multiclass classifier. Moreover, while the Kesler construction requires a sufficiently large number of features to generate the target class index, an abundance of features – obtained through any fixed linear or nonlinear transformation – can, in some cases, enhance classifier performance. Any form of nonlinear feature preprocessing may be beneficial. When the structure and parameters (weights) of this preprocessing layer are fixed, the remaining component can be a linear multiclass classifier with the desired accuracy and class sensitivities. Furthermore, when the hidden layer is large, applying regularization can simplify the classifier. Possible designs for the hidden layer include systematic polynomial expansion, kernel functions (via the kernel trick), a single layer of random sigmoidal perceptrons as in the RVFL network, or any nonlinear preprocessing motivated by the analysis of the learning task.

Ridge regression, also known as Tikhonov regularization, is a well-established technique that can create effective classifiers even in the presence of multicollinearity [17]. However, minimizing the L_2 norm of the weights is not a linear task and is therefore incompatible with LP solvers. To address this, techniques based on the L_1 norm – such as Lasso regression [41, 44] – have been successfully applied to both regression and classification problems, sometimes outperforming methods such as Partial Least Squares Discriminant Analysis [4, 39, 49]. One advantage of Ridge regularization is that quadratic penalization of the L_2 norm strongly biases the model toward smaller weights compared to the L_0 or L_1 norms. In contrast, L_1- or L_0-based classifiers may rely on a few features with extremely high weights, potentially leading to overfitting. Although the L_2 norm cannot be reformulated as an LP or MBP task, in our study we instead use the L_1 or L_0 norms combined with additional constraints on the weights. These external restrictions help overcome some limitations associated with using the L_0 or L_1 norms alone.

The L_1 norm is effective at reducing classifier complexity by approximately minimizing the number of nonzero weights, while the L_0 norm – which directly

counts the number of nonzero components – can be used for exact weight reduction. Both norms can also penalize compensatory variables in linearly inseparable datasets (i.e., imperfect learning), as they directly influence the accuracy and sensitivity of the classifier. However, neither norm inherently limits the maximum value of any individual weight.

Because the absolute values of weights vary with the problem, an additional technique for estimating and constraining their size is useful. First, the classification problem can be solved by minimizing the L_∞ norm of the weights; the resulting L_∞ value (or a small multiple of it) can then serve as a problem-dependent upper bound on the weights. This upper bound is used in conjunction with other regularization techniques, such as those based on the L_0 or L_1 norms. While the L_∞ norm alone is a basic regularization tool that constrains individual weights to remain near zero, it is generally less effective than the L_1 or L_0 norms. Nevertheless, imposing a maximum weight constraint enhances classifier robustness against outliers and can be effectively combined with other techniques that are amenable to LP or MBP formulations.

Various performance criteria can be used during classifier training, and these metrics are also employed during cross-validation on a separate verification set. In this paper, we distinguish between two types of learning. *Perfect learning* refers to training on a dataset that is linearly separable, resulting in a classifier with perfect accuracy (and unit class sensitivity) on the training set. This approach is based on the hypothesis that perfect training accuracy will translate into acceptable performance on the verification set. In contrast, *imperfect learning* involves constructing a classifier that achieves at least a desired level of class sensitivity, which is particularly useful for inseparable training sets (even after feature expansion). The underlying hypothesis is that relaxing the requirement for perfect class sensitivity during training may yield better overall performance on the verification set.

In this paper, we present a comprehensive guide for constructing a multiclass classifier through a series of LP or MBP tasks that implement a linear version of the Kesler construction for both perfect and imperfect learning. We then formulate various criteria for classifier robustness [37, 54] and sparsity [7, 30, 36] in both the original nonlinear form and an equivalent linear form based on the L_1 or L_0 norms with additional weight constraints. The resulting LP and MBP tasks vary in computational complexity and can be solved using various professional solvers or employed to test solver performance.

Section 2 reviews basic concepts about patterns, their sets, and one-vs-all multiclass classifiers, as well as their linear forms with unknown weights. Section 3 introduces the Kesler construction for formulating learning constraints under perfect and imperfect learning. Section 4 presents methods for converting the perfect

learning problem for linear multiclass classifiers into an LP task by exploiting the linearity of the Kesler constraints. The revisited simplex method is employed to enhance robustness and simplify the classifier. Section 5 discusses optimal classifier simplification via an advanced MBP formulation, along with imperfect learning tasks that target specific accuracy and critical sensitivity levels. Section 6 summarizes the implementation of the learning tasks and presents experimental results on three representative cases – the UCI Iris Flower, UCI Wine, and UCI Seeds datasets – followed by ablation studies in Section 7. These datasets were obtained from the UCI Machine Learning Repository [28]. Finally, Section 8 provides a discussion, and Section 9 concludes the paper by summarizing the key findings.

2. Multiclass classifier primer

Let $m, n, H \in \mathbb{N}$ be the number of patterns (samples), variables (features), and classes, respectively, satisfying $m \geqslant H \geqslant 2$. The multiclass classifier [19, 31] is a function $\mathrm{c} : \mathbb{R}^n \to \{0, 1, \ldots, H\}$, where $\mathrm{c}(\mathbf{x}) = 0$ indicates an unknown class. A pattern is a pair $(\mathbf{x}, c)$, where $\mathbf{x} \in \mathbb{R}^n$ and $c \in \{1, \ldots, H\}$. The pattern set $\mathcal{S} = \{(\mathbf{x}_i, c_i^*) : i = 1, \ldots, m\}$ can be represented by matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ and vector $\mathbf{c}^* \in \{1, \ldots, H\}^m$. The set $\mathcal{S}$ can be divided into H classes:

$$C_k = \{(\mathbf{x}_i, c_i^*) : c_i^* = k\}, \quad k = 1, \ldots, H. \tag{1}$$

The corresponding cardinality vector $\mathbf{m} \in \mathbb{N}^H$ has components:

$$m_k = \operatorname{card} C_k = \sum_{i=1}^{m} \left(1 - \operatorname{sign}(|c_i^* - k|)\right), \quad k = 1, \ldots, H, \tag{2}$$

where sign is the signum function satisfying $\operatorname{sign}(0) = 0, \operatorname{sign}(z) = 1, \forall z > 0$. We accept only classes represented by at least one pattern.

The general majority classifier of $\mathbf{x}$ evaluates:

$$y_k = \mathrm{f}(\mathbf{x}, \mathbf{w}_k), \quad k = 1, \ldots, H, \tag{3}$$

where $\mathbf{w}_k \in \mathbb{R}^{n+1}$ is a weight vector for the class C_k and f is a scoring function that gives a score for given input $\mathbf{x}$ and weights $\mathbf{w}_k$. The multiclass classifier output is determined by the majority principle:

$$\mathrm{c}(\mathbf{x}) \in \operatorname*{argmax}_{k=1,\ldots,H} y_k. \tag{4}$$

where sometimes, to increase the differences, the $softmax$ function is applied to y_k before application of $argmax$. However, when the maximum argument is not

unique, we set $\mathrm{c}(\mathbf{x}) = 0$. Focusing on linear optimization tasks, we use the simple linear model:

$$\mathrm{f}(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{n} w_j x_j \tag{5}$$

with $x_0 = 1$.

The classifier performance can be measured using various approaches which are based on the comparison of desired class memberships c_i^* with the classifier outputs $c_i = \mathrm{c}(\mathbf{x}_i)$ for $i = 1, \ldots, m$. The i-th pattern is correctly classified when $c_i = c_i^*$. The simple performance measure is the *accuracy* acc as the relative frequency of correctly classified patterns in the pattern set $\mathcal{S}$. Being focused on the individual classes, we define the *class sensitivity* se_k as the relative frequency of correctly classified patterns within class C_k. The *critical sensitivity* se_{crit} is the minimal value from the class sensitivities as the most stringent criterion for classifier performance.

3. Kesler construction for perfect and imperfect learning

The perfect learning technique, introduced first, is based on forming and solving a system of inequalities. The corresponding Kesler construction [16, 20] is designed as follows: each pattern generates $H - 1$ inequalities. Therefore, we form a set

$$\mathcal{D}_i = \{1, \ldots, H\} \setminus \{c_i^*\}, \quad i = 1, \ldots, m \tag{6}$$

and define general Kesler majority conditions

$$\mathrm{f}(\mathbf{x}_i, \mathbf{w}_{c_i^*}) - \mathrm{f}(\mathbf{x}_i, \mathbf{w}_k) \geqslant 1 \tag{7}$$

for all pairs (i, k) where $i \in \{1, \ldots, m\}$, $k \in \mathcal{D}_i$ according to (6).

In the case of a linear classifier, we obtain linear inequalities

$$\sum_{j=0}^{n} (w_{c_i^*,j} - w_{k,j}) x_{i,j} \geqslant 1 \tag{8}$$

for all pairs (i, k) where $i \in \{1, \ldots, m\}$, $k \in \mathcal{D}_i$. The bias uncertainty is addressed by the additive condition

$$\sum_{k=1}^{H} w_{k,0} = 0. \tag{9}$$

The introduction of artificial variables $\xi_{i,k} \geqslant 0$ allows for diminishing the separation power for $i = 1, \ldots, m$, $k \in \mathcal{D}_i$. The resulting inequality system is

$$\mathrm{f}(\mathbf{x}_i, \mathbf{w}_{c_i^*}) - \mathrm{f}(\mathbf{x}_i, \mathbf{w}_k) \geqslant 1 - \xi_{i,k} \tag{10}$$

or equivalently

$$\sum_{j=0}^{n}(w_{c_i^*,j} - w_{k,j})x_{i,j} \geqslant 1 - \xi_{i,k} \tag{11}$$

with the same bias stabilization condition.

For linear multiclass classifiers, both perfect and imperfect learning may involve additional optimization constraints, such as:

- L_∞ weight norm constraint (13) or minimization (14) to enhance classifier robustness
- L_1 weight norm minimization (16, 17) to reduce classifier complexity
- L_1 slack norm minimization (20) or (22, 23) to increase classifier accuracy
- L_0 weight norm minimization (29, 30) to reduce classifier complexity
- L_0 slack norm minimization (36, 37) to maximize classifier accuracy
- L_0 slack norm minimization (41–43) to maximize classifier sensitivity

all of them specified in linear form to keep them suitable for optimization solvers. The referenced formulas are explained in Sections 4 and 5.

4. (Im)perfect learning as a linear programming task

Both perfect and imperfect learning of a linear multiclass classifier must be also specified as linear optimization problems [13, 42, 45]. Therefore, additional requirements must be formulated as linear constraints or objective functions of $w_{k,j}$ and $\xi_{i,k}$. Well-known techniques for realizing the L_1 [47] and L_∞ [38] norms will also be employed.

4.1. Robustness of the classifier

The robustness to spiky noise [37, 54] in $\mathbf{x}$ is described by the criterion

$$Q_1 = \max_{\substack{k=1,\ldots,H \\ j=1,\ldots,n}} |w_{k,j}|, \tag{12}$$

as the L_∞ metric of active weight list.

Constraining the criterion $Q_1 \leqslant w^*$ is straightforward, involving a system of $2Hn$ fixed boundaries:

$$-w^* \leqslant w_{k,j} \leqslant w^*, \quad k = 1, \ldots, H, \ j = 1, \ldots, n. \tag{13}$$

However, if Q_1 is the subject of optimization, we must introduce $Q_1 \geqslant 0$ as a new variable, leading to inequalities [38]

$$-Q_1 \leqslant w_{k,j} \leqslant Q_1, \quad k = 1, \ldots, H, \ j = 1, \ldots, n, \tag{14}$$

and minimize the objective function Q_1.

4.2. Reduction of classifier complexity

Classifier complexity reduction [7, 30, 36] refers to the minimization of the number of nonzero weights, a topic to be explored in the next section. Alternatively, we can approximate the classifier simplification task by employing the L_1 metric of active weights to produce the criterion

$$Q_2 = \sum_{k=1}^{H} \sum_{j=1}^{n} |w_{k,j}|, \tag{15}$$

which can be minimized or constrained. The linear form of Q_2 is achieved using new variables $t_{k,j} \geqslant 0$ for $k = 1, \ldots, H$ and $j = 1, \ldots, n$. In this case,

$$Q_2 = \sum_{k=1}^{H} \sum_{j=1}^{n} t_{k,j}, \tag{16}$$

where

$$-t_{k,j} \leqslant w_{k,j} \leqslant t_{k,j}, \quad k = 1, \ldots, H, \ j = 1, \ldots, n. \tag{17}$$

We then minimize the objective function Q_2 or set the constraint $Q_2 \leqslant q^*$, respectively.

Another approximation focuses on reducing occurrences of explanatory variables. Analogically to (16, 17), we can design adequate criterion Q_3 in the linear form using new variables $u_j \geqslant 0$ for $j = 1, \ldots, n$ as

$$Q_3 = \sum_{j=1}^{n} u_j, \tag{18}$$

where

$$-u_j \leqslant w_{k,j} \leqslant u_j, \quad k = 1, \ldots, H, \ j = 1, \ldots, n. \tag{19}$$

We then minimize the objective function Q_3 or set the constraint $Q_3 \leqslant u^*$.

4.3. Simple accuracy control

The imperfection of a classifier can be approximately managed by minimizing or constraining the criterion

$$Q_4 = \sum_{i=1}^{m} \sum_{k \in \mathcal{D}_i} \xi_{i,k}, \tag{20}$$

which represents the linear form of the L_1 [47] metric for non-negative components.

We can also construct the criterion

$$Q_5 = \sum_{i=1}^{m} \max_{k \in \mathcal{D}_i} \xi_{i,k}, \tag{21}$$

which penalizes individual patterns. Using variables $v_i \geqslant 0$ for $i = 1, \ldots, m$, the linear form is

$$Q_5 = \sum_{i=1}^{m} v_i, \tag{22}$$

where

$$0 \leqslant \xi_{i,k} \leqslant v_i, \quad i = 1, \ldots, m, \ k \in \mathcal{D}_i. \tag{23}$$

Penalization can also be applied to individual classes as

$$C_L = \sum_{i: c_i^* = L} v_i, \quad L = 1, \ldots, H. \tag{24}$$

5. (Im)perfect learning as mixed binary programming task

Binary variables complicate any linear programming task and can lead to NP-complexity in classifier learning. However, they enable the precise calculation of the number of nonzero weights and the number of misclassified patterns. Thus, the complexity of the classifier and class sensitivities can be exactly optimized. A basic technique [50] can be illustrated with the elementary task of sign(z) minimization for $z \geqslant 0$. The optimal solution is $z = 0$.

Introducing a binary variable $b \in \{0, 1\}$, we can construct the Mixed Binary Programming (MBP) task [34]

$$b = \min, \tag{25}$$

subject to

$$z \geqslant 0, \tag{26}$$

$$z \leqslant M \cdot b, \tag{27}$$

where $M > 0$ is any upper bound of z.

5.1. Minimization of classifier complexity

An analogue of the Q_2 criterion in the style of L_0 [12] metrics can be designed as

$$Q_6 = \sum_{k=1}^{H} \sum_{j=1}^{n} \text{sign}(|w_{k,j}|). \tag{28}$$

The linear form of Q_6 is

$$Q_6 = \sum_{k=1}^{H} \sum_{j=1}^{n} t_{k,j}^{+}, \text{ where } t_{k,j}^{+} \in \{0, 1\}, \tag{29}$$

$$-M \cdot t_{k,j}^{+} \leqslant w_{k,j} \leqslant M \cdot t_{k,j}^{+}, \quad k = 1, \ldots, H, \ j = 1, \ldots, n, \tag{30}$$

with a sufficiently large $M > 0$.

Minimizing Q_6 creates a classifier with the minimum number of active connections, i.e., low complexity. An alternative approach to classifier simplification focuses on reducing occurrences of explanatory variables. An improvement on the Q_3 criterion in the style of L_0 is

$$Q_7 = \sum_{j=1}^{n} \max_{k=1,\ldots,H} \operatorname{sign}(|w_{k,j}|). \tag{31}$$

The linear form of Q_7 is

$$Q_7 = \sum_{j=1}^{n} u_j^{+}, \quad u_j^{+} \in \{0, 1\}, \tag{32}$$

$$-M \cdot u_j^{+} \leqslant w_{k,j} \leqslant M \cdot u_j^{+}, \quad k = 1, \ldots, H, \ j = 1, \ldots, n. \tag{33}$$

5.2. Exact accuracy control

The accuracy [48] of classification, defined as

$$acc = 1 - err = 1 - \frac{1}{m} \sum_{i=1}^{m} \max_{k \in \mathcal{D}_i} \operatorname{sign}(\xi_{i,k}), \tag{34}$$

is the relative number of correctly classified patterns. Thus, maximizing accuracy is equivalent to minimizing the classification error (err), i.e., minimizing criterion

$$Q_8 = \sum_{i=1}^{m} \max_{k \in \mathcal{D}_i} \operatorname{sign}(\xi_{i,k}). \tag{35}$$

The linear form of Q_8 is

$$Q_8 = \sum_{i=1}^{m} d_i, \quad d_i \in \{0, 1\} \tag{36}$$

$$0 \leqslant \xi_{i,k} \leqslant M d_i, \quad i = 1, \ldots, m, \ k \in \mathcal{D}_i. \tag{37}$$

The criterion Q_8 precisely represents the number of misclassified patterns. Utilizing the same artificial variables, we can count misclassifications within individual classes as

$$S_L = \sum_{i:c_i^*=L} d_i, \quad L = 1, \ldots, H. \tag{38}$$

5.3. Sensitivity-based classifier

Maximizing accuracy does not necessarily focus on learning errors within individual pattern classes, which can lead to error disproportions, a major argument against accuracy maximization. For a system of H classes $C_1, \ldots, C_H$ consisting of $m_1, \ldots, m_H$ patterns and with $S_1, \ldots, S_H$ learning errors, class sensitivities [24] are defined as

$$se_L = 1 - \frac{S_L}{m_L} \in [0, 1], \quad L = 1, \ldots, H. \tag{39}$$

Maximizing se_L is equivalent to minimizing S_L/m_L, and a constraint $se_L \geqslant se_L^*$ is equivalent to $S_L/m_L \leqslant 1 - se_L^*$ for any desired class sensitivity se_L^*. Maximizing critical sensitivity [23]

$$se_{\text{crit}} = \min_{L=1,\ldots,H} se_L \tag{40}$$

can lead to uniformity of learning errors across the classes. Equivalently, we minimize $1 - se_{\text{crit}}$, leading to the criterion

$$Q_9 = \max_{L=1,\ldots,H} \left(\frac{S_L}{m_L} \right) \tag{41}$$

and the corresponding optimization task is

$$Q_9 = \min, \tag{42}$$

$$\frac{S_L}{m_L} \leqslant Q_9, \quad L = 1, \ldots, H. \tag{43}$$

Alternatively, we can also constrain the critical sensitivity $se_{\text{crit}} \geqslant se_{\text{crit}}^*$ by conditions $S_L/m_L \leqslant 1 - se_{\text{crit}}^*$ for all classes.

6. Experimental part

The time complexity of LP and MBP tasks is a critical factor influencing the practical applicability of the proposed approach. We selected five optimization tasks for implementation and testing. Their relationships to metrics, objective functions, and constraints are summarized in Table 1. The tasks were implemented

in MATLAB R2021a [32], using the Optimization Toolbox and the GUROBI 9.5.2 solver [6], running on a Cisco UCS C220 M5 server with two Intel Xeon Gold 6154 processors, 384 GB RAM, and a virtualized Windows 10 64-bit operating system. For GUROBI solver, time limit of 500 s was specified to each solved task.

Table 1: Suggested multiclass classifier learning tasks

task	learning	metric	method	mini-mize	constraints
T1	perfect	L_∞	LP	Q_1	
T2	perfect	L_1	LP	Q_2	$Q_1 \leqslant w^*$
T3	perfect	L_0	MBP	Q_6	$Q_1 \leqslant w^*$
T4	imperfect	L_1	MBP	Q_2	$Q_1 \leqslant w^*$, $Q_8 \leqslant (1 - acc^*)m$, $S_L \leqslant (1 - se^*_{\text{crit}})m_L$, $\forall L$
T5	imperfect	L_0	MBP	Q_6	$Q_1 \leqslant w^*$, $Q_8 \leqslant (1 - acc^*)m$, $S_L \leqslant (1 - se^*_{\text{crit}})m_L$, $\forall L$

The revised simplex method [14] is strongly preferred for LP tasks due to its generation of vertex solutions when the LP task has degenerate solutions. In contrast, the CPU time consumption in MBP tasks heavily depends on optimization heuristics and their settings. Therefore, tasks T3-T5 can also be used for benchmarking of various optimization solvers.

In all cases, the accuracy of resulting classifiers was estimated using leave-one-out method cross-validation [55]. For other properties, L_∞, L_0, L_1, and se_{crit} results in tables are reported as additional information and were computed for classifier retrained on the whole dataset. For other implemented methods tested on the datasets, Lasso Regression and Ridge Regression for classification task using the one-vs-all scheme were implemented in MATLAB using functions from Statistics and Machine Learning Toolbox. The results were then confirmed by implementation in R for Windows (ver. 4.3.1, [35]) using packages *caret* (ver. 6.0.94) and *glmnet* (ver. 4.1.8, [44]). Accuracy was determined using the "best" rule for $\lambda \in \{0.0001, 0.0002, \ldots, 0.001, 0.002, \ldots, 0.01, 0.02, \ldots, 10\}$. The other two reference methods were k-nearest neighbor classifier with parameter $k \in \{2, 3, \ldots, 10\}$ and Logistic Regression with set of 11 logarithmically-spaced regularization strengths λ from 10^{-6} through $10^{-0.5}$. These methods for classification task using the one-vs-all scheme were implemented in MATLAB using functions from Statistics and Machine Learning Toolbox. The last reference method was linear SVM with parameter $C \in \{1, 10, 100, 1000, \infty\}$, implemented in R for Windows using package *kernlab* (ver. 0.9.33), and the results confirmed through the implementation in MATLAB with Statistics and Machine Learning Toolbox.

6.1. Iris Flower classification

The well-known UCI Iris Flower dataset [9, 15] consists of 150 four-dimensional patterns of three classes. Direct application of perfect learning is impossible because the second and third classes are not linearly separable. In this case, a quadratic transformation was employed to obtain extended patterns of length 14, consisting of all $x_j, x_j^2, x_j x_k$ for $j < k$. The resulting linear multiclass classifier has 42 active weights of 14 features and three biases.

The robustness to spiky noise can be ensured by L_∞ minimization (task T1) or by constraining the L_∞ metric in tasks T2 and T3. The goal was to obtain a sparse multiclass classifier with the best possible accuracy and/or critical sensitivity. In the first experiment, $w^* = 10^6$ was used as an upper estimate for L_∞, instead of $w^* \to +\infty$ for the unlimited optimization. The results of tasks T1–T3 are summarized in Table 2. In the case of L_∞ minimization (T1), robustness is guaranteed, but the number of active weights is maximized with no classifier complexity reduction. Conversely, L_0 minimization (T3) leaves only four active weights, demonstrating significant classifier simplification and the least robustness. The resulting linear multiclass classifier is based solely on four nonlinear features: $x_2^2, x_3^2, x_4^2, x_3 x_4$. This case also showed the best cross-validation values for accuracy and critical sensitivity. L_1 minimization (T2) simplified the multiclass classifier to nine active weights.

Table 2: Perfect learning of Iris Flower multiclass classifier with unlimited weights

task	acc	se_{crit}	L_∞	L_1	L_0
T1	0.9200	0.8800	**24.8531**	920.6110	42
T2	0.9200	0.8600	87.9723	**243.1309**	10
T3	**0.9600**	**0.9400**	1000000	1773069.13	**4**

The second experiment was focused on L_1 minimization (T2) under various values of the robustness constraint w^*. A compromise between robustness ($w^* = 25$) and weight number reduction ($w^* = 10^6$) was observed, as shown in Table 3. The best accuracy and critical sensitivity were achieved for $w^* = 100$.

The third experiment was focused on L_0 minimization (T3) under various values of w^*. A similar compromise behavior pattern was observed, as indicated in Table 4, although the best accuracy and critical sensitivity were achieved in several cases.

Imperfect learning techniques are not necessary for linearly separable patterns but can yield better multiclass classifier performance. Using a fixed robustness constraint $w^* = 50$, the role of the minimum required critical sensitivity $se^*_{\text{crit}} \in [0.95, 1.00]$ in tasks T4 and T5 was studied. The results of L_1 and L_0

Table 3: Perfect learning of Iris Flower multiclass classifier by L_1 minimization

w^*	acc	se_{crit}	L_1	L_0
25	0.9267	**0.9000**	917.4791	42
30	0.9267	**0.9000**	334.8309	17
40	0.9333	**0.9000**	247.2860	12
50	0.9333	**0.9000**	243.1309	11
60	0.9333	**0.9000**	243.1309	10
70	0.9267	0.8800	243.1309	10
80	0.9267	0.8800	243.1309	10
90	0.9267	0.8800	243.1309	9
100	**0.9400**	**0.9000**	243.1309	9
1500	0.9333	0.8800	243.1309	9
10^6	0.9333	0.8800	243.1309	9

Table 4: Perfect learning of Iris Flower multiclass classifier by L_0 minimization

w^*	acc	se_{crit}	L_1	L_0
25	0.9267	0.9000	917.4791	42
30	0.9333	0.9000	360.7763	16
40	0.9133	0.8800	264.4520	11
50	**0.9600**	**0.9400**	323.4474	9
60	0.9467	0.9200	276.9528	8
70	**0.9600**	**0.9400**	273.9668	7
80	0.9267	0.9000	318.6562	7
90	0.9333	0.8800	294.1427	7
100	0.9533	0.9200	331.2908	7
150	0.9333	0.9000	406.8887	5
1000	0.9533	0.9200	1653.847	5
1500	**0.9600**	**0.9400**	1823.415	4
10^6	**0.9600**	**0.9400**	1773069.13	4

minimizations are summarized in Tables 5 and 6. The best cross-validated critical sensitivity $se_{\text{crit}} = 0.98$ was obtained by L_1 minimization for $se^*_{\text{crit}} = 0.98$. However, the simplest possible multiclass classifier with only two active weights was obtained by L_0 minimization for $se^*_{\text{crit}} = 0.96$. This linear multiclass classifier is based only on two nonlinear features: x_1x_4, x_3x_4 and ignores the variable x_2 with a critical sensitivity of $se_{\text{crit}} = 0.94$.

For reference methods, the comparison of results is given in Table 7. The best results with accuracy $acc = 0.9867$ and critical sensitivity $se_{\text{crit}} = 0.98$ were obtained by L_1 minimization and imperfect learning.

Table 5: Imperfect learning of Iris Flower multiclass classifier by L_1 minimization for $w^* = 50$

se^*_{crit}	acc	se_{crit}	L_1	L_0
1.00	0.9333	0.9000	243.1309	11
0.99	0.9267	0.8800	243.1309	11
0.98	**0.9867**	**0.9800**	6.9650	8
0.97	0.9667	0.9400	6.9643	7
0.96	0.9667	0.9200	2.6859	6
0.95	0.9600	0.9200	2.6822	7

Table 6: Imperfect learning of Iris Flower multiclass classifier by L_0 minimization for $w^* = 50$

se^*_{crit}	acc	se_{crit}	L_1	L_0
1.00	0.9600	0.9400	323.4474	9
0.99	0.9333	0.9000	306.9399	9
0.98	**0.9667**	**0.9600**	122.2356	3
0.97	0.9333	0.9000	122.2356	3
0.96	**0.9600**	**0.9400**	71.7573	2
0.95	0.9400	0.9000	87.8788	2

Table 7: Comparison of results for Iris Flower multiclass classifier

task	acc	se_{crit}	se_1	se_2	se_3
L_∞	0.9267	0.9000	0.9800	0.9000	0.9000
L_1	**0.9867**	**0.9800**	1.0000	0.9800	0.9800
L_0	0.9667	0.9600	0.9800	0.9600	0.9600
Ridge	0.9600	0.9423	1.0000	0.9423	0.9792
Lasso	0.9800	0.9423	1.0000	0.9423	0.9791
kNN	0.9667	0.9400	1.0000	0.9400	0.9600
SVM	0.9800	0.9423	1.0000	0.9423	0.9791
Logistic regression	0.9800	0.9600	1.0000	0.9600	0.9800

6.2. Wine classification

The UCI Wine [2,3] dataset classifies wines based on chemical compositions. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines (e.g. alcohol, color intensity, hue, and flavonoids), which were centered and scaled prior to other analysis. The dataset includes 178 samples of wines with 59, 71, and 48 samples per class and the data were centered and scaled. While the dataset is linearly separable, all the experiments focused on perfect learning strategy.

First, the influence of the w^* value on accuracy (acc) and critical sensitivity (se_{crit}) in tasks T1, T2, and T3, focusing on perfect learning, was studied. A value

of $w^* = 10^6$ was used as an upper bound for L_∞ instead of $w^* \to +\infty$. The results of tasks T1, T2, and T3 are summarized in Table 8, along with leave-one-out accuracy and critical sensitivity.

The second experiment investigated the influence of the w^* value on minimization. The values of w^* were chosen as small multiples ℓ of the L_∞ value. Multiple values of $\ell \in \{1, 1.1, 1.2, \ldots, 2.9, 3, 4, 5, 6, 7, 8, 9, 10\}$ were tested and the best ℓ that achieved the highest accuracy was selected. Outstanding results were obtained already for $w^* = 1.1 \cdot L_\infty$, with an accuracy of 0.9719 and a critical sensitivity of 0.9577 achieved (see Table 9).

Table 8: Perfect learning of Wine multiclass classifier with unlimited weights

task	acc	se_{crit}	L_∞	L_1	L_0
T1	**0.9607**	**0.9437**	**0.5369**	16.1609	39
T2	0.9551	0.9296	1.6379	**9.2568**	17
T3	0.9494	0.9155	1000000	3466003.94	**5**

Table 9: The influence of w^* by L_1 minimization in the Wine dataset

ℓ	$w^* = \ell \cdot L_\infty$	acc	se_{crit}	L_∞	L_1	L_0
1	0.5369	**0.9719**	**0.9577**	0.5370	13.3757	34
1.1	0.5906	**0.9719**	**0.9577**	0.5906	11.5679	29
1.2	0.6443	**0.9719**	0.9437	0.6443	10.6952	26
1.3	0.6980	0.9663	0.9437	0.6980	10.1602	21
1.4	0.7517	0.9607	0.9437	0.7517	9.9618	21
1.5	0.8054	0.9607	0.9437	0.8054	9.7765	21
1.6	0.8591	0.9663	0.9437	0.8591	9.6182	21
1.7	0.9128	0.9607	0.9375	0.9128	9.5190	21
1.8	0.9665	0.9607	0.9375	0.9665	9.4732	20
1.9	1.0202	0.9551	0.9375	1.0202	9.4522	20
2	1.0739	0.9551	0.9375	1.0739	9.4316	19
2.1	1.1276	0.9551	0.9375	1.1276	9.4118	19
2.2	1.1813	0.9494	0.9296	1.1813	9.3921	19
2.3	1.2350	0.9551	0.9296	1.2350	9.3723	19
2.4	1.2887	0.9551	0.9296	1.2887	9.3532	19
2.5	1.3424	0.9551	0.9296	1.3424	9.3350	19
2.6	1.3961	0.9551	0.9296	1.3961	9.3168	19
2.7	1.4498	0.9551	0.9296	1.4498	9.3005	19
2.8	1.5035	0.9551	0.9296	1.5035	9.2847	19
2.9	1.5572	0.9551	0.9296	1.5572	9.2695	18
3	1.6108	0.9551	0.9296	1.6108	9.2588	18
4	2.1478	0.9551	0.9296	1.6379	9.2568	17
10	5.3695	0.9551	0.9296	1.6379	9.2568	17

In the final experiment, the influence of w^* on L_0 minimization was studied. The value of w^* was set in the same way as in the previous case. The results are summarized in Table 10. The best accuracy (0.9888) with a critical sensitivity of 0.9718 was achieved for only 10 active weights. For the first cultivar, important values of alkalinity of ash, flavonoids, and proline were identified. For the second cultivar, alcohol, ash, nonflavonoid phenols, and hue were found to be important. For the third cultivar, important values of flavonoids, color intensity, and OD280/OD315 of diluted wines were determined to be important.

Table 10: The influence of w^* by L_0 minimization in the Wine dataset

ℓ	$w^* = \ell \cdot L_\infty$	acc	se_{crit}	L_∞	L_1	L_0
1	0.5369	0.9775	0.9583	0.5370	13.3757	34
1.1	0.5906	0.9607	0.9437	0.5906	12.1338	25
1.2	0.6443	0.9663	0.9296	0.6443	12.0022	22
1.3	0.6980	0.9663	0.9437	0.6980	11.4720	20
1.4	0.7517	0.9831	**0.9718**	0.7517	11.0389	17
1.5	0.8054	0.9719	0.9577	0.8054	11.7200	16
1.6	0.8591	0.9775	0.9583	0.8591	10.8465	15
1.7	0.9128	0.9775	0.9583	0.9128	11.3090	14
1.8	0.9665	0.9719	0.9375	0.9665	11.8004	14
1.9	1.0202	0.9719	0.9577	1.0202	11.4365	13
2	1.0739	0.9551	0.9375	1.0739	12.2657	13
2.1	1.1276	0.9494	0.9167	1.1276	12.6212	13
2.2	1.1813	0.9551	0.9437	1.1813	12.3159	12
2.3	1.2350	0.9719	0.9661	1.2350	11.8711	12
2.4	1.2887	0.9831	**0.9718**	1.2887	11.1482	11
2.5	1.3424	0.9607	0.9577	1.3424	12.8001	11
2.6	1.3961	0.9663	0.9437	1.3961	12.8404	11
2.7	1.4498	0.9831	**0.9718**	1.4498	11.8183	10
2.8	1.5035	**0.9888**	**0.9718**	1.5035	12.8682	10
2.9	1.5572	0.9663	0.9577	1.5572	13.3285	10
3	1.6108	0.9719	0.9661	1.6108	14.0299	10
4	2.1478	0.9607	0.9577	2.1478	15.5642	9
5	2.6847	0.9719	0.9577	2.6847	20.4239	8
6	3.2217	0.9719	0.9577	3.2217	20.8891	8
7	3.7586	0.9663	0.9492	3.7586	19.3737	8
8	4.2956	0.9494	0.9296	4.2956	22.1794	7
9	4.8325	0.9438	0.9153	4.8325	23.9084	7
10	5.3695	0.9382	0.9155	5.3695	23.9816	7

The comparison of all results for the Wine multiclass classifier is given in Table 11. The best results with accuracy $acc = 0.9888$ and critical sensitivity $se_{\text{crit}} = 0.9718$ were obtained by L_0 minimization, by Ridge regression, and also by Logistic regression.

Table 11: Comparison of results for Wine multiclass classifier

task	acc	se_{crit}	se_1	se_2	se_3
L_∞	0.9607	0.9437	0.9831	0.9437	0.9583
L_1	0.9719	0.9577	0.9831	0.9577	0.9792
L_0	**0.9888**	**0.9718**	1.0000	0.9718	1.0000
Ridge	**0.9888**	**0.9718**	1.0000	0.9718	1.0000
Lasso	0.9775	0.9795	0.9833	1.0000	0.9795
kNN	0.9719	0.9296	1.0000	0.9296	1.0000
SVM	0.9550	0.9796	0.9833	1.0000	0.9796
Logistic regression	**0.9888**	**0.9718**	1.0000	0.9718	1.0000

6.3. Seeds classification

The UCI Seeds [10,11] dataset originated from the properties of three different varieties of wheat such as Kama, Rosa, and Canadian. Each category of wheat has 70 elements which were randomly selected for the experiment. The dataset contains the properties of the images which were recorded on X-ray KODAK plates. Seven features are described as area, perimeter, compactness, length and width of kernel, asymmetry coefficient, and length of kernel groove. Because of the linear non-separability of patterns with the use of the original seven features, after centering and scaling a quadratic transformation was employed to obtain extended patterns of length 35, consisting of all $x_j, x_j^2, x_j x_k$ for $j < k$. First, the influence of the w^* value on accuracy (acc) and critical sensitivity (se_{crit}) in tasks T1, T2, and T3, focusing on perfect learning, was studied. A value of $w^* = 10^6$ was used as an upper bound for L_∞ instead of $w^* \to +\infty$. The results of tasks T1, T2, and T3 are summarized in Table 12.

The second experiment investigated the influence of the w^* value on minimization. The values of w^* were chosen as small multiples of the L_∞ value. Multiple

Table 12: Perfect learning of Seeds multiclass classifier with unlimited weights

task	acc	se_{crit}	L_∞	L_1	L_0
T1	**0.9524**	0.9286	**1.4364**	128.2417	105
T2	**0.9524**	0.9286	8.5926	**48.9160**	28
T3	**0.9524**	**0.9429**	1000000	2708786.28	**6**

values of $\ell \in \{1, 1.1, 1.2, \ldots, 2.9, 3, 4, 5, 6, 7, 8, 9, 10\}$ were tested. The results for various numbers of components are shown in Table 13. Outstanding results were obtained for $w^* = 1.7 \cdot L_\infty$, where an accuracy of 0.9810 and a critical sensitivity of 0.9571 were achieved.

Table 13: The influence of w^* by L_1 minimization in the Seeds dataset

ℓ	$w^* = \ell \cdot L_\infty$	acc	se_{crit}	L_∞	L_1	L_0
1	1.4364	0.9571	0.9286	1.4364	124.0297	100
1.1	1.5801	0.9619	0.9286	1.5801	78.3972	63
1.2	1.7237	0.9429	0.8857	1.7237	71.2188	54
1.3	1.8674	0.9571	0.9143	1.8674	65.8275	49
1.4	2.0110	0.9619	0.9286	2.0110	62.1566	46
1.5	2.1546	0.9571	0.9143	2.1546	59.3862	43
1.6	2.2983	0.9714	0.9286	2.2983	56.9387	42
1.7	2.4419	**0.9810**	**0.9571**	2.4419	54.7155	41
1.8	2.5856	0.9714	0.9429	2.5856	53.1868	38
1.9	2.7292	0.9619	0.9286	2.7292	52.3158	36
2	2.8729	0.9619	0.9286	2.8729	51.8568	32
2.1	3.0165	0.9571	0.9143	3.0165	51.5089	32
2.2	3.1602	0.9571	0.9143	3.1602	51.1700	33
2.3	3.3038	0.9524	0.9143	3.3038	50.8488	31
2.4	3.4474	0.9524	0.9143	3.4474	50.5660	32
2.5	3.5911	0.9524	0.9143	3.5911	50.3212	32
2.6	3.7347	0.9524	0.9143	3.7347	50.0870	32
2.7	3.8784	0.9524	0.9143	3.8784	49.8740	31
2.8	4.0220	0.9619	0.9286	4.0220	49.7147	30
2.9	4.1657	0.9619	0.9286	4.1657	49.5821	30
3	4.3093	0.9571	0.9143	4.3093	49.4712	30
4	5.7457	0.9429	0.9000	5.7457	49.0846	30
5	7.1822	0.9381	0.9143	7.1822	48.9772	29
6	8.6186	0.9524	0.9286	8.5926	48.9160	28
10	14.3643	0.9524	0.9286	8.5926	48.9160	28

In the final experiment, the influence of w^* on L_0 minimization was studied. The value of w^* was set in the same way as in the previous case. The results are summarized in Table 14. The best accuracy (0.9762) with a critical sensitivity of 0.9571 was achieved for only 19 active weights.

The comparison of all results for the Seeds multiclass classifier is given in Table 15. The best results with accuracy $acc = 0.9810$ and critical sensitivity $se_{\text{crit}} = 0.9571$ were obtained by L_1 minimization.

Table 14: The influence of w^* by L_0 minimization in the Seeds dataset

ℓ	$w^* = \ell \cdot L_\infty$	acc	se_{crit}	L_∞	L_1	L_0
1	1.4364	0.9619	0.9286	1.4364	124.0297	100
1.1	1.5801	0.9714	0.9429	1.5801	81.9812	56
1.2	1.7237	0.9524	0.9143	1.7237	73.9773	47
1.3	1.8674	0.9429	0.8857	1.8674	71.2270	41
1.4	2.0110	0.9571	0.9143	2.0110	65.9422	37
1.5	2.1546	0.9619	0.9286	2.1546	63.4570	34
1.6	2.2983	0.9667	0.9429	2.2983	65.8132	32
1.7	2.4419	0.9524	0.9000	2.4419	63.5372	29
1.8	2.5856	0.9571	0.9143	2.5856	63.6289	27
1.9	2.7292	0.9524	0.9286	2.7292	55.7340	26
2	2.8729	**0.9762**	0.9429	2.8729	58.1767	24
2.1	3.0165	0.9714	**0.9571**	3.0165	56.1830	22
2.2	3.1602	0.9619	0.9429	3.1602	61.0581	21
2.3	3.3038	0.9667	0.9286	3.3038	59.7557	21
2.4	3.4474	0.9667	0.9429	3.4474	62.4732	19
2.5	3.5911	**0.9762**	**0.9571**	3.5911	58.0470	19
2.6	3.7347	0.9667	0.9286	3.7347	61.0221	18
2.7	3.8784	**0.9762**	0.9429	3.8784	62.5230	18
2.8	4.0220	**0.9762**	0.9429	4.0220	63.0583	17
2.9	4.1657	0.9571	0.9286	4.1657	61.8498	17
3	4.3093	0.9619	0.9000	4.3093	64.2894	17
4	5.7457	0.9524	0.9000	5.7457	64.8710	14
5	7.1822	0.9524	0.9143	7.1822	61.2110	13
6	8.6186	0.9571	0.9000	8.6186	67.6689	11
7	10.0550	0.9714	0.9429	10.0550	68.9269	10
8	11.4915	0.9714	**0.9571**	11.4915	76.3873	9
9	12.9279	0.9524	0.9286	12.9279	88.5678	9
10	14.3643	0.9476	0.9000	14.3643	78.5011	9

Table 15: Comparison of results for Seeds multiclass classifier

task	acc	se_{crit}	se_1	se_2	se_3
L_∞	0.9524	0.9286	0.9286	0.9714	0.9571
L_1	**0.9810**	**0.9571**	0.9571	1.0000	0.9857
L_0	0.9762	**0.9571**	0.9571	1.0000	0.9714
Ridge	0.9238	**0.9571**	0.9571	0.9859	0.9853
Lasso	0.9619	**0.9571**	0.9571	0.9859	0.9853
kNN	0.9381	0.8857	0.8857	0.9571	0.9714
SVM	0.9667	**0.9571**	0.9571	0.9859	0.9853
Logistic regression	0.9762	**0.9571**	0.9571	1.0000	0.9714

7. Ablation studies

When quantifying the individual effects of the techniques, previous results indicate that both the L_1 and L_0 norms provide effective regularization, with no clear advantage of one over the other. However, when using the L_0 norm, one must solve at least MBP tasks, which are generally NP-hard, whereas the L_1 norm leads to LP tasks that have polynomial-time complexity. However, when imperfect learning is introduced, the resulting tasks also become MBP regardless of the norm used.

The impact of w^* is evident from several experiments focused on minimizing the L_1 or L_0 norms for various w^* values. To quantify the influence of Kesler construction on final accuracy or class sensitivity, an independent multiclass classifier employing either L_1 or L_0 regularization – with an added fixed weight constraint and a one-vs-all scheme – is required. When implemented in MATLAB or R for the L_1 norm, this corresponds to a Lasso-constrained regression. The results on the tested datasets are presented in Table 16, with the w^* values chosen as shown in Table 3 for the Iris dataset, Table 9 for the Wine dataset, and Table 13 for the Seeds dataset.

Table 16: Ablation studies for weight constraint w^* and the Kesler construction using Lasso

task	Iris Flower		Wine		Seeds	
	acc	se_{crit}	acc	se_{crit}	acc	se_{crit}
Lasso	**0.9800**	**0.9423**	0.9775	0.9795	0.9619	**0.9571**
Lasso-constrained	**0.9800**	**0.9423**	**0.9943**	**0.9795**	0.9667	**0.9571**
Lasso-constrained & Kesler	0.9400	0.9000	0.9719	0.9577	**0.9810**	**0.9571**

For the Iris Flower dataset with quadratic transformation, the best accuracy ($acc = 0.9800$) and critical sensitivity ($se_{\text{crit}} = 0.9423$) were achieved by both the original Lasso regression and the Lasso-constrained version, indicating that the weight constraint alone did not improve performance. In contrast, when the Kesler construction was applied with the L_1 norm and imperfect learning, the final accuracy improved ($acc = 0.9867$, $se_{\text{crit}} = 0.9800$). However, comparing the Lasso-constrained regression with the one-vs-all scheme to the classifier employing only the Kesler construction (i.e., using the L_1 norm with perfect learning, which yielded $acc = 0.9400$ and $se_{\text{crit}} = 0.9000$) indicates that the Kesler construction alone degrades performance.

For the Wine dataset without quadratic transformation, the Lasso-constrained regression achieved exceptionally high accuracy ($acc = 0.9943$), confirming the significant impact of w^*. In this case, adding the Kesler construction only

worsened the results. When more features are available for the Kesler construction, the target class index can be constructed more accurately; consequently, if a quadratic transformation is also applied to this dataset, the accuracy of the classifier with the Kesler construction improves slightly ($acc = 0.9775$). However, this combined approach still does not match the performance of the Lasso-constrained regression alone.

For the Seeds dataset with quadratic transformation, the classifier employing the Kesler construction outperformed the classifier without it in terms of accuracy ($acc = 0.9810$ versus $acc = 0.9667$), as well as outperforming the classifier that used neither the Kesler construction nor an individual weight constraint. In summary, the overall effect of Kesler construction varies depending on the dataset and the specific setup.

8. Discussion

Error penalization and weight regularization are fundamental tools in classifier training. We present a guide for constructing a linear classifier by transforming it into a LP or MBP task, which can then be solved using any LP or MBP optimization solver. This approach enables us to combine advanced regularization techniques – such as L_1 or L_0 minimization – with an additional maximum weight constraint to enhance robustness, and, if needed, to incorporate a class sensitivity constraint (imperfect learning).

We compared two linear optimization approaches: one based on L_0 norm penalization and regularization, which yields MBP tasks with NP-hard complexity, and one based on L_1 norm penalization and regularization, which results in LP tasks that can be solved in polynomial time.

The Iris Flower classification task allowed us to compare L_1 and L_0 minimization strategies, weight reduction and feature selection, as well as the effect of additional class sensitivity constraints (i.e., perfect vs. imperfect learning). The highest critical sensitivity (0.9800) and accuracy (0.9867) were achieved using L_1 minimization for imperfect learning, with a target sensitivity of $se^*_{\text{crit}} = 0.98$ and a weight constraint of $w^* = 50$. In contrast, perfect learning (i.e., without class sensitivity restrictions) yielded inferior results. The accuracy values obtained are generally comparable to published results for the Iris Flower dataset [15].

For the Wine dataset, which is linearly separable, we compared L_∞, L_1, and L_0 minimization under various perfect learning strategies. The highest accuracy (acc = 0.9888) with a critical sensitivity of $se_{\text{crit}} = 0.9718$ was achieved using L_0 minimization with a weight constraint of $w^* = 1.5035$, as well as with Ridge regression and Logistic regression.

For the Seeds dataset, the highest accuracy (0.9810) and critical sensitivity ($se_{\text{crit}} = 0.9571$) were achieved using L_1 minimization with a weight constraint of $w^* = 2.4419$, which results in a simple LP task.

Less complex learning techniques that employ a weakly motivated L_1 norm exhibit behavior similar to that of the more exact L_0 norm. This observation, supported by our examples, suggests that computing time during weight optimization can be reduced since only LP tasks need to be solved.

The value of w^* is also crucial for efficient classifier simplification and for maximizing the desired performance measure (e.g., accuracy). Based on results from several datasets, the final L_∞ value obtained from the L_∞-based classifier – and its low multiples – provides good estimates for a reasonable interval of w^* values. However, minimizing L_∞ alone is not particularly effective in terms of accuracy and sensitivity.

In this paper, we presented various optimization approaches for constructing linear multiclass classifiers. These approaches were summarized, unified, modified, and reformulated as LP or MBP tasks. After implementing the proposed tasks using MATLAB's Optimization Toolbox and the GUROBI solver, we conducted a numerical study on several multiclass classification problems. We found that acceptable critical sensitivity and accuracy can be achieved by minimizing the L_1 criterion under a weight constraint ($w^* < +\infty$), which leads to an LP task in the case of perfect learning. The resulting classifier is robust and utilizes a reduced number of weights, positively influencing its generalization ability. Moreover, this optimization process has only polynomial time complexity. In contrast, minimizing the L_0 norm or employing imperfect learning with a desired critical sensitivity results in MBP tasks, which, while potentially improving accuracy and sensitivity, are generally NP-hard and more time-consuming.

9. Conclusions

We have demonstrated that specifying a classifier using a system of inequalities – which incorporates both the Kesler construction and an effective classifier complexity reduction technique (such as minimizing the L_0 or L_1 norm with a maximum weight constraint) – is a viable alternative to current methods for creating classifiers. This system of inequalities can be solved independently using any external LP or MBP optimization solver and can achieve accuracies and class sensitivities comparable to widely used methods. Furthermore, when the L_1 norm is employed, the resulting task remains within polynomial-time complexity.

References

[1] O. Adigun and B. Kosko: Training deep neural classifiers with soft diamond regularizers, 2024, DOI: 10.48550/arXiv.2412.20724

[2] S. Aeberhard, D. Coomans, and O.Y. de Vel: Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, **27**, (1994), 1065–1077, DOI: 10.1016/0031-3203(94)90145-7

[3] S. Aeberhard and M. Forina: Wine, UCI Machine Learning Repository, 1991.

[4] M. Barker and W. Rayens: Partial least squares for discrimination. *Journal of Chemometrics*, **17**(3), (2003), 166–173, DOI: 10.1002/cem.785

[5] D. Bertsimas, A. King, and R. Mazumder: Best subset selection via a modern optimization lens. *arXiv: Methodology*, 2015, DOI: 10.48550/arXiv.1507.03133

[6] D. Bertsimas and B. Stellato: Online mixed-integer optimization in milliseconds. *INFORMS Journal on Computing*, **34**(4), (2022), 2229–2248, DOI: 10.48550/arXiv.1907.02206

[7] D. Blalock, J.J. Gonzalez Ortiz, J. Frankle, and J. Guttag: What is the state of neural network pruning? *Proceedings of machine learning and systems*, **2**, (2020), 129–146, DOI: 10.48550/arXiv.2003.03033

[8] G.D. Cavalcanti and R.J. Soares: Ranking-based instance selection for pattern classification. *Expert Systems with Applications*, **150**, (2020), 113269, DOI: 10.1016/j.eswa.2020.113269

[9] A. Chakraborty, N. Faujdar, A. Punhani, and S. Saraswat: Comparative study of k-means clustering using iris data set for various distances. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 332–335, IEEE, 2020, DOI: 10.1109/Confluence47617.2020.9058328

[10] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. Kowalski, and S. Lukasik: Seeds, UCI Machine Learning Repository, 2012.

[11] M. Charytanowicz, J. Niewczas, P. Kulczycki, P.A. Kowalski, S. Łukasik, and S. Żak: Complete gradient clustering algorithm for features analysis of x-ray images. In *Information technologies in biomedicine*, pages 15–24, Springer, 2010, DOI: 10.1007/978-3-642-13105-9_2

[12] A. Dedieu, H. Hazimeh, and R. Mazumder: Learning sparse classifiers: Continuous and mixed integer optimization perspectives. *The Journal of Machine Learning Research*, **22**(1), (2021), 6008–6054, DOI: 10.48550/arXiv.2001.06471

[13] S.-C. Fang and S. Puthenpura: *Linear optimization and extensions: theory and algorithms*. Prentice-Hall, Inc., 1993.

[14] F.A. Ficken: *The simplex method of linear programming*, Courier Dover Publications, 2015.

[15] R.A. Fisher: Iris. UCI Machine Learning Repository, 1988.

[16] V. Franc and V. Hlavac: Multi-class support vector machine. In *2002 International Conference on Pattern Recognition*, volume 2, pages 236–239. IEEE, 2002. DOI: 10.1109/ICPR.2002.1048282

[17] M. Gruber: *Improving Efficiency by Shrinkage: The James–Stein and Ridge Regression Estimators*. CRC Press, 1998, DOI: 10.1201/9780203751220

[18] L. Guo: Extreme learning machine with elastic net regularization. *Intelligent Automation & Soft Computing*, **26**(3), (2020), 421–427, DOI: 10.32604/iasc.2020.013918

[19] M.R. Gupta, S. Bengio, and J. Weston: Training highly multiclass classifiers. *The Journal of Machine Learning Research*, **15**(1), (2014), 1461–1492, DOI: 10.5555/2627435.2638582

[20] P.E. Hart, D.G. Stork, and R.O. Duda: *Pattern classification*, Wiley Hoboken, 2000.

[21] A.E. Hoerl and R.W. Kennard: Ridge regression: Applications to nonorthogonal problems. *Technometrics*, **12**(1), (1970), 69–82, DOI: 10.2307/1267352

[22] A.E. Hoerl and R.W. Kennard: Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, **12**(1), (1970), 55–67, DOI: 10.2307/1271436

[23] R. Hrebik and J. Kukal: Concept of hidden classes in pattern classification. *Artificial Intelligence Review*, pages 1–18, 2023, DOI: 10.1007/s10462-023-10430-6

[24] R. Hrebik, J. Kukal, and J. Jablonsky: Optimal unions of hidden classes. *Central European Journal of Operations Research*, **27**(1), (2019), 161–177, DOI: 10.1007/s10100-017-0496-5

[25] P. Huang, Z. Yang, W. Wang, and F. Zhang: Denoising low-rank discrimination based least squares regression for image classification. *Information Sciences*, **587**, (2022), 247–264, DOI: 10.1016/j.ins.2021.12.031

[26] M. Jiao, Y. Yang, D. Wang, and P. Gong: The conjugate gradient optimized regularized extreme learning machine for estimating state of charge. *Ionics*, **27**, (2021), 4839–4848, DOI: 10.1007/s11581-021-04169-9

[27] S. Kang, S. Cho, and P. Kang: Constructing a multi-class classifier using one-against-one approach with different binary classifiers. *Neurocomputing*, **149**, (2015), 677–682, DOI: 10.1016/j.neucom.2014.08.006

[28] M. Kelly, R. Longjohn, and K. Nottingham: The UCI machine learning repository, 2023, URL: https://archive.ics.uci.edu

[29] Z. Kolter and A. Madry: Adversarial robustness – theory and practice, 2018, (Accessed on April 25, 2024), URL: https://adversarial-ml-tutorial.org/

[30] A. Lazarevic and Z. Obradovic: Effective pruning of neural network classifier ensembles. In *IJCNN'01. International Joint Conference on Neural Networks, Proceedings (Cat. No. 01CH37222)*, volume 2, pages 796–801, IEEE, 2001, DOI: 10.1109/IJCNN.2001.939461

[31] Y. Liu, Z. You, and L. Cao: A novel and quick SVM-based multi-class classifier. *Pattern Recognition*, **39**(11), (2006), 2258–2264, DOI: 10.1016/j.patcog.2006.05.034

[32] C. Lopez: *MATLAB optimization techniques*, Apress, 2014, DOI: 10.1007/978-1-4842-0292-0

[33] A.K. Malik, R. Gao, M. Ganaie, M. Tanveer, and P.N. Suganthan: Random vector functional link network: recent developments, applications, and future directions. *Applied Soft Computing*, page 110377, (2023), DOI: 10.1016/j.asoc.2023.110377

[34] Y.-S. Niu, Y. You, and W.-Z. Liu: Parallel DC cutting plane algorithms for mixed binary linear program. In *World Congress on Global Optimization*, pages 330–340, Springer, 2019, DOI: 10.1007/978-3-030-21803-4_34

[35] R Core Team: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023, URL: https://www.R-project.org/

[36] R. Reed: Pruning algorithms – a survey. *IEEE transactions on Neural Networks*, **4**(5), (1993), 740–747, DOI: 10.1109/72.248452

[37] L. Rice, E. Wong, and Z. Kolter: Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104, PMLR, 2020, DOI: 10.48550/arXiv.2002.11569

[38] S.A. Ruzinsky and E.T. Olsen: L_1 and L_∞ minimization via a variant of Karmarkar's algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**(2), (1989), 245–253, DOI: 10.1109/29.21687

[39] L. Rysova, P. Cejnar, O. Hanus, V. Legarova, J. Havlik, H. Nejeschlebova, I. Nemeckova, R. Jedelska, and M. Bozik: Use of maldi-tof ms technology to evaluate adulteration of small ruminant milk with raw bovine milk. *Journal of Dairy Science*, **105**(6), (2022), 4882–4894, DOI: 10.3168/jds.2021-21396

[40] A.M.E. Saleh, M. Arashi, and B.G. Kibria: *Theory of ridge regression estimation with applications*. John Wiley & Sons, 2019, DOI: 10.1002/9781118644478

[41] F. Santosa and W.W. Symes: Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, **7**(4), (1986), 1307–1330, DOI: 10.1137/0907087

[42] A. Schrijver: *Theory of linear and integer programming*, John Wiley & Sons, 1998.

[43] B. Schölkopf and A.J. Smola: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001, DOI: 10.7551/mitpress/4175.001.0001

[44] R. Tibshirani: Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**(1), (1996), 267–288, DOI: 10.1111/j.2517-6161.1996.tb02080.x

[45] R.J. Vanderbei et al.: *Linear programming*. Springer, 2020, DOI: 10.1007/978-3-030-39415-8

[46] S. Vargaftik and Y. Ben-Itzhak: Efficient multiclass classification with duet. In *Proceedings of the 2nd European Workshop on Machine Learning and Systems*, EuroMLSys '22, pages 10–19, Association for Computing Machinery, 2022, DOI: 10.1145/3517207.3526970

[47] P. Venkataraman: *Applied optimization with MATLAB programming*, John Wiley & Sons, 2009.

[48] S.W. Wilson: Classifier fitness based on accuracy. *Evolutionary computation*, **3**(2), (1995), 149–175, DOI: 10.1162/evco.1995.3.2.149

[49] S. Wold, M. Sjöström, and L. Eriksson: PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, **58**(2), (2001), 109–130, PLS Methods, DOI: 10.1016/S0169-7439(01)00155-1

[50] L.A. Wolsey: *Integer programming*. John Wiley & Sons, 2020, DOI: 10.1002/9781119606475

[51] H. Yildirim and M.R. Özkale: An enhanced extreme learning machine based on Liu regression, *Neural Processing Letters*, **52**(1), (2020), 421–442, DOI: 10.1007/s11063-020-10263-2

[52] H. Yildirim and M.R. Özkale: LL-ELM: A regularized extreme learning machine based on L_1-norm and Liu estimator. *Neural Computing and Applications*, **33**(16), (2021), 10469–10484, DOI: 10.1007/s00521-021-05806-0

[53] H. Yildirim and M.R. Özkale: A combination of ridge and Liu regressions for extreme learning machine. *Soft Computing*, **27**(5), (2023), 2493–2508, DOI: 10.1007/s00500-022-07745-x

[54] B. Zhang, D. Jiang, D. He, and L. Wang: Rethinking Lipschitz neural networks and certified robustness: A boolean function perspective. *Advances in Neural Information Processing Systems*, **35**, (2022), 19398–19413, DOI: 10.48550/arXiv.2210.01787

[55] J. Zhang and S. Wang: A fast leave-one-out cross-validation for SVM-like family. *Neural Computing and Applications*, **27**, (2016), 1717–1730, DOI: 10.1007/s00521-015-1970-4

[56] S. Zheng and C. Ding: A group lasso based sparse KNN classifier. *Pattern Recognition Letters*, **131**(1), (2020), 227–233, DOI: 10.1016/j.patrec.2019.12.020