Proceedings of the 2nd World Congress on Electrical Engineering and Computer Systems and Science (EECSS'16)
Budapest, Hungary - August 16 - 17, 2016
Paper No. MVML 104
DOI: 10.11159/mvml16.104

Feature Selection with a Budget

Matthias Richter[1], Georg Maier[2], Robin Gruna[3], Thomas Längle[4], and Jürgen Beyerer[5]
[1,5]Karlsruhe Institute of Technology
Adenauerring 4, 76131 Karlsruhe, Germany
matthias.richter@kit.edu; juergen.beyerer@kit.edu
[1,2,3,4,5]Fraunhofer IOSB
Fraunhoferstr. 1, 76131 Karlsruhe, Germany
georg.maier@iosb.fraunhofer.de; robin.gruna@iosb.fraunhofer.de; thomas.laengle@iosb.fraunhofer.de

Abstract - Feature selection is an important step in all practical applications of pattern recognition. As such, it is not surprising that during the past decades it has received a lot of attention from the research community. The topic is well understood and many methods have been put to the test. Most methods, however, overlook an aspect critical to real-time applications: limited computation time. The set of selected features must not only be suitable to solve the task, but must also ensure that the task can be solved within the available time. With this in mind, we propose a method for feature selection with a budget. We approach the problem by stating feature selection as a multi-objective optimization problem. This problem is solved using the well known NSGA-II algorithm. We evaluate our approach using one synthetic and two real-world datasets. We explore the properties of the genetic algorithm and investigate the classification performance of the resulting selections. Our results show that the selected feature sets are highly suitable, especially when considering real-time systems.

Keywords: Feature selection, multi-objective, evolutionary algorithm, pattern recognition, real-time systems, visual inspection

1. Introduction

The selection of informative features is an important step in any pattern recognition system. On the one hand, it reduces model complexity and allows users to interpret the system more easily. On the other hand, it often improves prediction performance: it is well known that too many features have adversarial effects in that regard [10]. Systematic approaches to feature selection (often also called *variable selection*) can be traced back to the 1970s [1, 22, 16] and the community has since produced a large arsenal of different methods, each with their own advantages and drawbacks. A review of these methods is out of scope of this paper; interested readers are instead referred to the recent summary by Chandrashekar and Sahin [4].

For the purpose of this paper, the feature selection problem is formalized as follows: Given a set $\mathbf{T} = \{(\mathbf{x}_n, y_n) | n = 1..N\}$ of N training samples $\mathbf{x}_n$ and associated labels y_n and a set $\mathbf{F} = \{\phi_d(\cdot) | d = 1..D\}$ of D feature extractors $\phi_d(\cdot)$ that map training samples to scalar features[1], the goal is to select a subset $\mathbf{S}^\star \subseteq \mathbf{F}$ so that some utility function $U(\mathbf{S}, \mathbf{T})$ is maximized:

$$\mathbf{S}^\star = \arg\max_{\mathbf{S}} U(\mathbf{S}, \mathbf{T}). \tag{1}$$

Different approaches to feature selection may be distinguished in the way $\mathbf{S}$ is constructed. For example, features may be selected by iteratively adding informative features, iteratively removing non-informative features, or by repeated random sampling. More importantly, however, methods differ in how $U(\mathbf{S}, \mathbf{T})$ is calculated. Correlation based feature selection (CBFS) [8], for example, defines the objective as

$$U_{\text{CBFS}}(\mathbf{S}, \mathbf{T}) = \frac{k\overline{r_{y\mathbf{S}}}}{\sqrt{k + k(k-1)\overline{r_{\mathbf{SS}}}}}, \tag{2}$$

where $k = |\mathbf{S}|$ is the number of selected features, $\overline{r_{y\mathbf{S}}}$ denotes the mean correlation of class labels and features, and $\overline{r_{\mathbf{SS}}}$ denotes ~~the mean correlation between the selected features~~.

[1] A *feature* describes some property of a particular object, while a *feature extractor* is the method to do so. However, we (as many others) use the terms interchangeably when the meaning is clear from the context.

Equation (2) illustrates a common theme in most (if not all) feature selection methods: the utility function is sensitive to the number of selected features. More precisely, the aim is to minimize the number of selected features, while keeping their utility at a maximum. This is reasonable, as more features mean that a more complex and hence more error-prone classifier is needed. In practical applications, however, one is typically not interested in the smallest number of most discriminative features, but instead in *the most discriminative features that require the least computational effort*. It is often implicitly assumed that the two are the same, but they are not: If a selection $\mathbf{S}_a$ and a smaller selection $\mathbf{S}_b$ provide the same discriminative power, but $\mathbf{S}_a$ is faster to compute than $\mathbf{S}_b$, one should prefer $\mathbf{S}_a$ over $\mathbf{S}_b$. However, most feature selection methods will favour $\mathbf{S}_b$ instead.

This aspect is especially relevant in real-time applications such as human computer interaction, robotics, and automated visual inspection. The latter serves as the main motivation for our work. In visual inspection, especially in sorting of bulk material, it can even be acceptable to sacrifice prediction performance in order to reduce the computational cost and meet the deadline when the system is under heavy load. In such a scenario, it may also be favourable to take the variance of computation cost into account. Consider, for example, canonical features such as color moments, compactness or point symmetry. The time required to compute these features directly depends on the size of the objects. If one expects to encounter objects of vastly different sizes, such features should be sparingly selected to reduce the risk of missing a deadline.

1.1. Contributions

In this paper, we propose a feature selection method that takes the computational cost of each feature into account. We formalize the method as a multi-objective optimization problem: maximizing utility while minimizing the expected computational cost. We employ the well known genetic algorithm NSGA-II [5] (non-dominant sorting genetic algorithm) to approximate the set of Pareto optimal solutions, i.e. solutions that dominate all other solutions with respect to at least one objective. Given this set of solutions, it is up to the user to decide upon a suitable set of features. We suggest a method to aid this decision. We apply our method to two bulk-good sorting scenarios as well as an augmented synthetic dataset from the 2003 NIPS feature selection challenge [7]. Where other methods either return just a single set of features or disregard their computational cost altogether, our approach yields a more dense representation of the whole Pareto-front of optimal solutions. To our best knowledge, this is the first time this has been done.

2. Related Work

Early genetic algorithm (GA) based approaches to the feature selection problem include [21] and [25]. In the latter, Yang and Honavar consider a wrapper approach based on an inter-pattern distance-based neural network learning algorithm. Two criteria, namely the accuracy of the neural network classifier and the cost of the feature selection are combined into a single objective. The authors motivate feature cost by the examples of financial expense as well as risks associated with corresponding medical procedures required to obtain a specific feature, but do not make the connection to computation time. In their experiments, they demonstrate that the combined fitness function outperforms a fitness function that only considers the accuracy of the feature subset. In [11], Iswandy and Koenig also consider the costs of selected features. They select features using both GA and particle swarm optimization (PSO) and show that the latter has better convergence characteristics. Like Yang and Honavar, however, they combine the two goals into a single objective function instead of performing direct multi-objective optimization. Paclík et al. consider a group-wise forward selection, where features are added to the selection one after another [17]. When features are added, they do not only consider the performance gain, but also the additional time needed to compute the feature. Their criterion effectively favours cheap features that provide a small performance gain over better, but more expensive ones. Plasberg and Kleijn follow a similar route and introduce a complexity term into forward feature selection based on mutual information [18]. They determine optimal feature sets for different trade-offs of cost and utility in a grid search and use the selection that maximizes the mutual information between selection and labels.

Common to all these approaches is that they optimize a single objective function despite pursuing multiple optimization goals. That is not to say that multi-objective optimization has not been utilized in the context of feature selection. Morita et al., for example, utilize NSGA in the context of handwritten word recognition [15]. They define two objectives, where the first one maximizes the separation of clusters in the feature space, while the second one minimizes the number of selected features. Similarly, Hamdani et al. use NSGA-II to minimize classification error of a nearest neighbor (1-NN) classifier as well as the number of features [9]. The same approach is used by Tekgüç et al. in the context of facial expression recognition [23]. However, instead of minimizing the classification error of a 1-NN classier, they maximize the separation of classes as measured by the Fisher criterion. Xue et al. compare different multi-objective optimization methods including GA and PSO techniques in the context of feature selection [24]. The optimization goals are again to maximize the performance of a classifier and to minimize the number of selected features. Recently, Saroj proposed three criteria to select features: information gain, non-redundancy, and size of the selection [20]. The author independently optimizes all three goals using NSGA-II and compares the result against a GA that optimizes a single objective function that averages over all three criteria. The results of the analysis are inconclusive. While all of these methods utilize multi-objective optimization, none takes computation time into account.

Genetic algorithms have also been used to learn—as opposed to select—features. Ebner uses Cartesian Genetic Programming to evolve image processing routines to detect moving objects [6]. Processing pipelines are assembled from primitive operations implemented on a GPU so that the output images encode the likely position of moving objects. The fitness of a pipeline is assessed according to the distance of the detected position and the ground truth. In the context of automated visual inspection, Burla et al. apply genetic programming to obtain image processing procedures that are represented by processing graphs [3]. The output images are binarized and treated as encoding location of surface defects. The fitness of a processing pipeline is computed from defect detection performance on a validation set. Lillywhite et al. use a similar method to derive processing pipelines for general object detection [13]. None of these methods take computation time of the processing pipeline into account.

3. Methods

As mentioned in the introduction, our goal is to find sets of features that maximize utility and minimize computational cost. In order to do the latter, we define $c(\phi_d(\mathbf{x}_n))$ to be the cost for feature ϕ_d extracted from $\mathbf{x}_n$. The cost can be measured (e.g., computation time in ticks) or specified in advance (e.g., monetary costs to obtain the feature). We then compute the empirical mean and standard deviation of the costs over the dataset,

$$\bar{c}(\phi_d) = \frac{1}{N}\sum_{n=1}^{N} c(\phi_d(\mathbf{x}_n)), \quad \text{and} \quad \sigma_c(\phi_d) = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\left(c(\phi_d(\mathbf{x}_n)) - \bar{c}(\phi_d)\right)^2}.$$

To simplify the notation, we collect the mean costs and cost scatter in vectors, $\bar{\mathbf{c}} = \left(\bar{c}(\phi_1), \ldots, \bar{c}(\phi_D)\right)^\top$ (σ_c likewise). Similarly, a selection $\mathbf{S}$ is represented by the vector

$$\mathbf{s} = \left(\mathbf{1}_{\mathbf{S}}(\phi_1), \ldots, \mathbf{1}_{\mathbf{S}}(\phi_D)\right)^\top \in \{0,1\}^D, \tag{3}$$

where $\mathbf{1}_{\mathbf{S}}(\phi)$ indicates the membership $\phi \in \mathbf{S}$. Now, a selection criterion can be written as

$$U(\mathbf{S}, \mathbf{T}) = u(\mathbf{s}; \mathbf{T}) - \lambda \mathbf{s}^\top \bar{\mathbf{c}} - \mu \mathbf{s}^\top \sigma_c, \tag{4}$$

where $u(\mathbf{s}; \mathbf{T}) \in \mathbb{R}$ denotes the utility of $\mathbf{S}$ *without regard to the number of features*. We will define suitable utility functions in section 3.3. Note that this formulation includes "conventional" feature selection methods: If all costs are the same, the scatter term will be 0 and eq. (4) will amount to selecting the minimal number of most discriminative features.

An issue that already surfaced in section 2 is how λ and μ should be balanced. As the answer depends on the goal—fast prediction or low risks—as well as the choice of utility function, we can (and do) not provide a general answer. Instead, we opt to present the user with a set of possible solutions to choose from. We compute this set of possible solutions using the multi-objective optimization algorithm NSGA-II.

3.1. NSGA-II

Before moving on to feature selection, we briefly introduce NSGA-II. A detailed discussion of the algorithm and its properties can be found in the original paper [5]. NSGA-II is a genetic algorithm to minimize a vectorial objective function. An initial population of solution candidates is randomly sampled and ordered according to their *fitness*. By recombining solutions from the initial population and slightly changing *(mutating)* the newly created individuals, the population is expanded. The population is again sorted by fitness and only the fittest individuals are retained. The process is repeated until a stopping criterion is met. More formally, let $\mathbf{P}_t = \{\mathbf{s}_n | n = 1..N\}$ denote the population of solutions at step t and let $\mathbf{s}_p$ and $\mathbf{s}_q$ denote two *parent* solutions. The *offspring* is created as $\mathbf{s}_o = \text{mutate}\left(\text{crossover}\left(\mathbf{s}_p, \mathbf{s}_q\right)\right)$. How to select parents and carry out crossover and mutation is not further specified. The offspring and $\mathbf{P}_t$ are collected into an intermediate population, $\mathbf{O}_t$, and a fitness function f is used to define a (partial) ordering $\mathbf{s}_i <_f \mathbf{s}_k$ on $\mathbf{O}_t$. The top N solutions of $\mathbf{O}_t$ are kept as the new population $\mathbf{P}_{t+1}$.

The key to NSGA-II is how this ordering is constructed. Deb et al. suggest two criteria: *nondomination rank* and *crowding distance* [5]. The nondomination rank of a solution $\mathbf{s}$ is computed using the domination count: the number of other solutions $\mathbf{s}_n$ that dominate $\mathbf{s}$ with respect to at least one optimization-goal. The solutions with domination count 0 are assigned the nondomination rank 1. Next, the count of all solutions dominated by solutions with rank 1 is reduced by the number of such solutions. Solutions with updated domination count 0 are assigned the rank 2. The process is repeated with increasing nondomination ranks until all solutions are ranked. The nondomination rank effectively encodes the distance to the Pareto front: solutions with rank 1 reside on the Pareto front, solutions with rank 2 are one step removed from the Pareto front, etc.

Nondomination rank alone would be sufficient to induce an ordering of $\mathbf{P}$. To encourage diverse populations, however, solutions of equal rank are additionally sorted according to their crowding distance. The crowding distance of $\mathbf{s}$ is the perimeter the cuboid spanned by the nearest solutions that have the same rank as $\mathbf{s}$. Therefore, the crowding distance is inversely related to the local density around $\mathbf{s}$. NSGA-II assigns a higher fitness to solutions with a higher crowding distance—solutions in locally sparse regions—than to solutions with a smaller crowding distance.

3.2. Genetic Encoding and Genetic Operations

Encoding a selection is straightforward: the "genome" of a solution is the selection vector from eq. (3). Crossover and mutation are canonical: In the former, the entries of a new individual are randomly sampled from either parent with equal probability. The latter flips each entry in the offspring with probability $1/D$. Parents are selected by tournament selection: For each parent, two candidates are selected at random and the one with better fitness is chosen as parent solution. We stop the genetic algorithm after a maximum number of populations has been computed.

The objective function covers the same criteria as eq. (4): maximization of the selection's utility and minimization of the expected cost and cost scatter. As NSGA-II minimizes its objective, the objective function becomes

$$\omega(\mathbf{s}) = \left(-u(\mathbf{s};\mathbf{T}), \mathbf{s}^\top\bar{\mathbf{c}}, \mathbf{s}^\top\sigma_c\right)^\top. \tag{5}$$

Note that the cost-criteria make the assumption that the cost of a feature is independent of the selection. This is, of course, a simplification. In reality, some features are computed from other features (e.g., color moments, principal axis length, ...) and need less computation time if these features are already selected. For the sake of simplicity, however, we assume that the impact of this dependence is negligible.

3.3. Utility functions

What remains open is the choice of utility function. Here, we explore a filter and wrapper approach to feature selection.

Filter. The filter approach is inspired by CBFS [8], that is, we wish to select features that show high correlation with the labels. Unlike CBFS, however, we compute the multiple correlation coefficient (e.g., [12], pp. 225–227) between labels and the selected features instead of the mean correlation between labels and features. With $R_{\mathbf{SS}}$ (where $r_{ik} = \rho_{\phi_i\phi_k}$) denoting the matrix of pairwise correlation coefficients between the selected features and $\mathbf{r}_{\mathbf{S}y} = (\rho_{\phi_1 y}, \dots, \rho_{\phi_{|\mathbf{S}|}y})^\top$ denoting the vector of correlation coefficients between labels y and the features in $\mathbf{S}$, the multiple correlation coefficient can be calculated as

$$u_{\text{mcorr}}(\mathbf{s};\mathbf{T}) = \sqrt{R^2} = \sqrt{\mathbf{r}_{\mathbf{S},y}^\top R_{\mathbf{SS}}^{-1}\mathbf{r}_{\mathbf{S},y}}. \tag{6}$$

Note that, unlike other filter approaches, this criterion does not discourage redundancy of selected features. In our framework, this aspect is covered by the other optimization goals in eq. (5). In fact it might even be favourable to select correlated features, provided that these features increase the overall prediction performance without requiring too much additional computation time.

Wrapper. Wrapper methods determine the utility of a selection using the performance of a predictor trained on the selected features. Usually this means high computational load, since the predictor has to be first trained and then evaluated, possibly in cross-validation. In a genetic algorithm this is prohibitive, because the utility has to be computed for every individual in the current population. However, if we choose (multiclass) LDA as classifier, we can skip the evaluation and use the LDA criterion as proxy for the classification performance instead,

$$u_{\text{LDA}}(\mathbf{s};\mathbf{T}) = J(W) = Tr\left\{\left(WS_WW^\top\right)^{-1}\left(WS_BW^\top\right)\right\}. \tag{7}$$

Here, S_W denotes the within-class scatter matrix of the selected features, S_B denotes the between-class scatter matrix of the selected features, and W is the projection matrix that maximizes the LDA criterion (see, e.g., [2], pp. 186–192 for more details). A similar approach was used by Richter and Beyerer [19], although only in a two-class scenario.

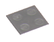
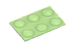
4. Experiments

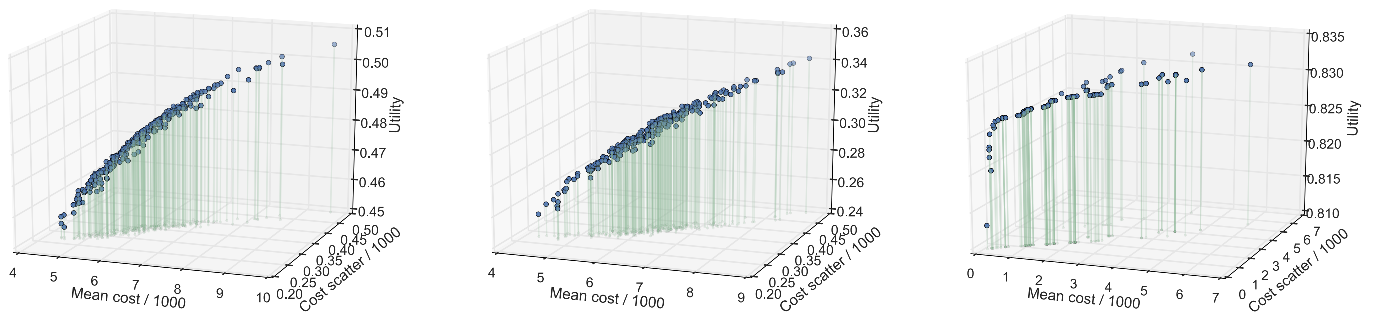
In the following, we describe our experiments to evaluate our method. We used three datasets for validation. One dataset is synthetic, while the two other represent real world sorting problems.

4.1. Synthetic Data

The 2003 NIPS feature selection challenge provided five datasets to evaluate feature selection algorithms [7]. The artificial *Madelon* dataset consists of 2 600 samples and 500 features per sample. The dataset was constructed in a way that no single feature is informative on its own. As this dataset does not include feature costs, we augmented each feature instance by randomly sampling a cost. The costs were drawn from a normal distribution $N(\mu_\phi, \sigma_\phi)$ where the parameters μ_ϕ and σ_ϕ were uniformly sampled ($\mu_\phi \sim U(0,100)$ and $\sigma_\phi \sim U(0.01,5)$) for each individual feature ϕ. Note that we did not take into account whether the feature was informative or not when sampling the costs. This means that uninformative features may be favoured by a selection that minimizes costs alongside utility.

Table 1: Overview of the datasets. In the lego dataset, all classes except class 8 contain bricks of different colors.

Dataset	**Class**	**Description**	**Sample Count**	**Example Image**
Lego	1	2×2 Lego bricks	380	
Lego	2	2×3 Lego bricks	330	
Lego	3	2×4 Lego bricks	438	
Lego	4	Lego Technic cross-profile axles of different sizes	468	
Lego	5	Lego Technic bricks of sizes 1×2 to 1×16	371	
Lego	6	Lego bricks of size 1×1 to 1×8	367	
Lego	7	Lego Technic connector bricks	415	
Lego	8	Lego Technic connector bricks of two different lengths	366	
Stones	1	Round, gray pebble stones with dark spots	1212	
Stones	2	Blue-gray pebble stones with light spots	3418	



(a) Madelon, 500 generations 250 individuals, mcorr utility.

(b) Madelon, 500 generations, 250 individuals, LDA utility.

(c) Lego, 100 generations, 150 individuals, mcorr utility.

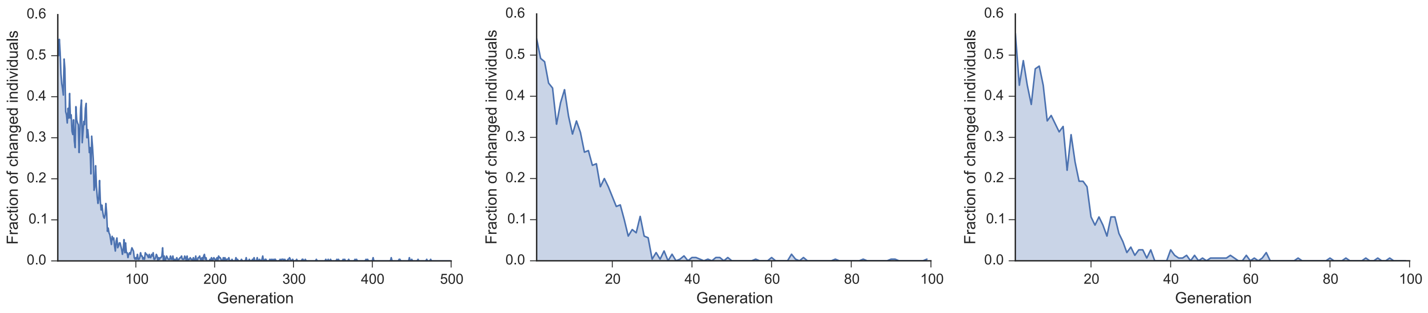
Fig. 1: Final population in the objective space. With the synthetic dataset solutions spread, while with real datasets they tend to cluster.

4.2. Real-world Data

Additionally to the synthetic data, we created two datasets representing bulk material sorting problems. For this purpose, an existing sorting system consisting of two line scan cameras was used to record images of Lego bricks and pebble stones. A total of 44 standard features—geometric features such as area, compactness, length and width, as well as color features like color moments of multiple orders—were extracted for each object in the images. The computation time required for calculating the features was also recorded. Consequently, in both datasets each sample is described by the assigned class label as well as a feature vector of length 44 as well as 44 associated costs in terms of required computation time.

The first dataset was obtained using various Lego bricks and hence will be referred to as “Lego” in the following. It includes eight classes of different bricks as depicted in Table 1 and consists of a total of 3 135 samples. This dataset serves as a prototype for sorting problems that involve man-made, manufactured objects with moderate intra-class variance but high similarity between (some) classes. Such tasks are challenging because under some viewing angles different objects may look the same. For instance, class 5 and 6 may only be discriminated from a side view, whereas class 1, 2 and 3 look the same when viewed from the (short) side.

For the second dataset, images of two different kinds of pebble stones were recorded and will be denoted as “Stones” in the remainder of this paper. It consists of two classes of stones and a total of 4 630 samples as also depicted in Table 1. In contrast to the Lego dataset, this dataset represents natural bulk material, where the intra-class variance is high, but classes can be more easily discriminated. As such, this dataset is representative for challenges on the other side of the spectrum. This ensures that our method is not only suitable for one type of sorting problems.



(a) Madelon, 250 individuals, LDA utility. (b) Lego, 250 individuals, mcorr utility. (c) Stones, 150 individuals, mcorr utility.

Fig. 3: Population change per generation. After large initial changes, the algorithm converges quickly. Note the different scales.

4.3. Properties of the genetic algorithm

To verify our method, we ran NSGA-II with a varying population size and for a varying number of generations. Fig. 1a and Fig. 1b show the final population of a run on the Madelon dataset. The mcorr-criterion (eq. (6)) produces similar results as the LDA-criterion (eq. (7)) in the sense that the distribution of mean cost and cost scatter is similar. In this dataset, higher utility of a selection means higher computation time and more cost scatter. There is no clear best solution nor are there any visible clusters. Note, however, that the dataset was artificially created and the costs were randomly sampled, so this result is to be expected. With real data, the situation presents itself differently. Fig. 1c shows the 100th generation of a run on the Lego dataset. While there is no clear “best” solution, one can make out several distinct clusters corresponding to different mean costs. Interestingly, most clusters do not show large variety in utility. In contrast to the Madelon dataset, the relationship between mean cost and cost scatter is not linear. Similar observations can be made with the Stones dataset (not shown).

Fig. 3 shows the relative change in population for all datasets. With both synthetic and real-world data, the algorithm quickly stabilizes. However, with the Madelon dataset, convergence takes considerably longer than with the the real-worlds datasets. This may be caused by the vastly larger number of features (500 in Madelon, 44 in Lego and Stones), but can also be attributed to the clustering property of the real-world datasets (cf. fig. 1c). The quick convergence is confirmed in fig. 2—after generation 30 the utility is not significantly changed until generation 60, where an inferior solution was replaced. Again, with the synthetic dataset (not shown) the convergence is slower and less pronounced.

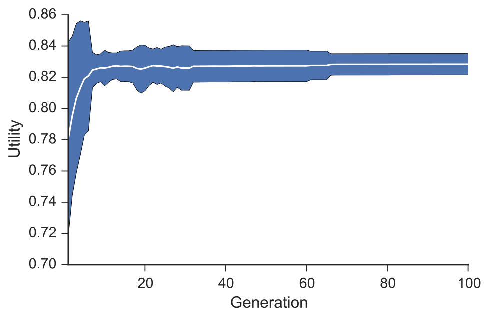
4.4. Goodness of selection

We now turn our attention to the selected features. We evaluated the goodness of each selection in a stratified 10 fold cross-validation, where we trained a SVM classifier with RBF kernel on the selected features. We did not perform any parameter tuning, but normalized the features to zero mean and unit variance prior to the training.

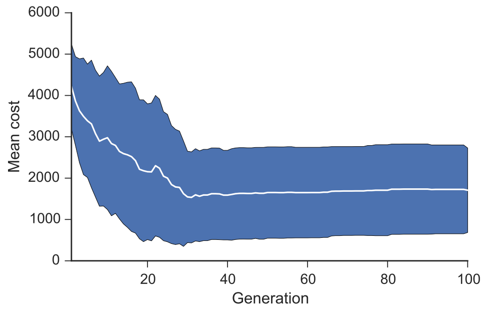
We used Matthews Correlation Coefficient (MCC, [14]) as a metric. In a two-class setting, it can be interpreted as the correlation between ground truth and prediction:

$$\mathrm{MCC} = \frac{\mathrm{TP}\cdot\mathrm{TN} - \mathrm{FP}\cdot\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}. \qquad (8)$$

Here TP $(\mathrm{TN}, \mathrm{FP}, \mathrm{FN})$ denote the number of true positive (true negative, false positive, false negative) classifications. To adapt MCC to a multi-class setting, we computed the metric for each class individually and averaged over the results.



(a) Utility of the whole population.



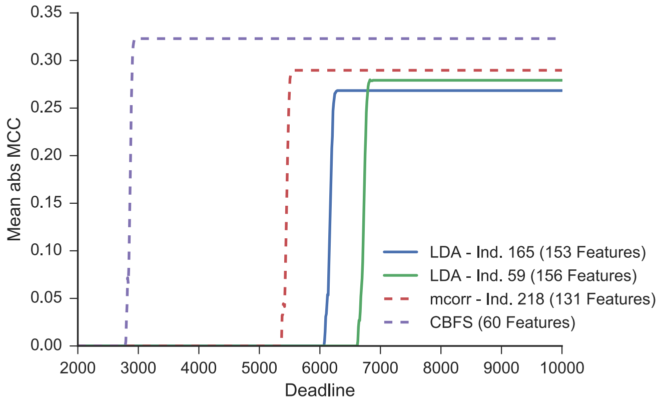
(b) Mean cost of the whole population.

Fig. 2: Change in objective function with the Lego dataset (250 individuals, mcorr utility). The white line shows the mean over the population while the shaded area shows the 2σ interval.

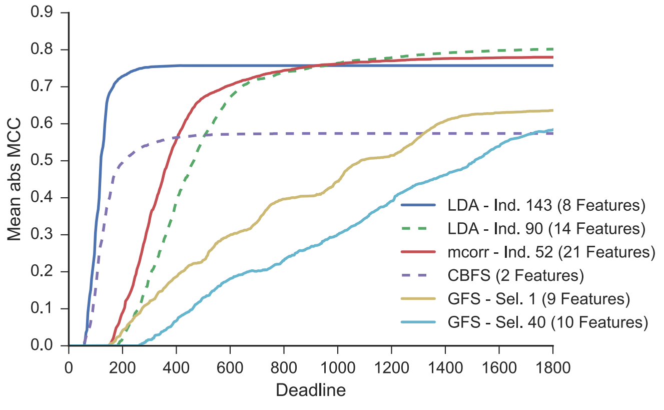
As we are interested in real-time applications, more specifically the sorting of bulk material, we simulated varying deadlines. For each deadline, we counted the number of misses and changed the prediction label of missed samples to an additional "missed" class. We compared our method against CBFS (eq. (2)) with greedy forward-selection. Features were selected exhaustively until all features were selected, but only the selection with highest utility was kept. As CBFS does not take computation time into account, this comparison is not really fair, but serves to illustrate the point that good selections are not necessarily good in terms of computation time. Additionally, we compared our method against GFS [17] on the Lego dataset (without feature grouping—grouping information was not available in the datasets). Again, multiple solutions corresponding to different deadlines were computed, but unlike CBFS, all solutions were considered in the evaluation.

The results are summarized in fig. 4. With the Madelon dataset, all solutions selected by our method (three of them shown in fig. 4a) perform worse than the baseline. However, none of the solutions, including the baseline, perform particularly well. It is noteworthy that with all solutions the mean MCC is 0, but almost instantly saturates when some deadline threshold is exceeded. This threshold as well as the peak performance is related to the number of selected features: more features result in worse performance and a larger deadline threshold. We attribute this behaviour, which is foreshadowed in fig. 1, to our random sampling of costs and the designed difficulties in selecting informative features. With the real-world datasets, CBFS is outperformed in both maximum prediction performance as well as computation speed. Especially in fig. 4c it can be seen that a small number of features does not automatically result in good real-time characteristics. Even though the selection with highest peak MCC (Individual 103, LDA criterion) contains more features as the baseline selection, the selection is much faster to compute. Similar observations can be made in fig. 4b. Here, it can be seen that one should favor a fast feature set (LDA individual 143) when the deadline is tight, but should switch to a different set when the conditions are more relaxed. It can also be seen that our method outperforms GFS, which, unlike CBFS, takes computation time into account. The first GFS selection also shows a step-like performance curve, indicating that certain discriminate, but expensive features become "active" once enough time is available to compute them.

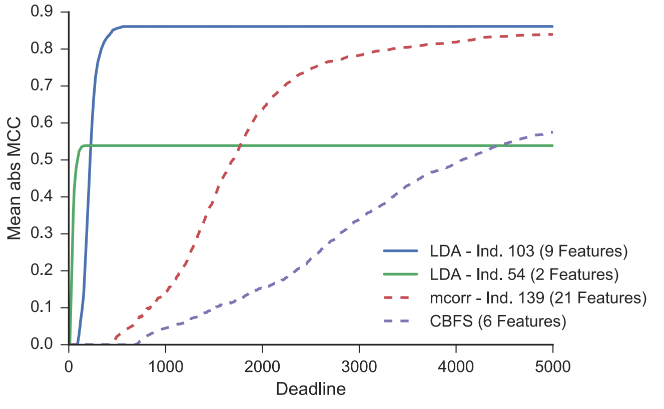
From our experiments, it is not clear whether the filter approach (mcorr utility) or the wrapper approach (LDA utility) yield overall better results. Which is to be preferred depends on the data set. However, with both the Lego and the Stones datasets, it can be seen that certain solutions are to be preferred under certain real-time conditions. Such an analysis of classifier performance against simulated deadlines can be used to aid the user in the selection of a suitable feature set for certain situations. It can also be used to automate this decision.



(a) Madelon dataset.



(b) Lego dataset.



(c) Stones dataset.

Fig. 4: Mean abs MCC vs. deadline. The labels indicate the selection method (e.g., LDA), the identifier of the individual (if applicable) and the number of selected features (best viewed in color).

5. Conclusion

In this paper, we presented a novel method for multi-objective feature selection with genetic algorithms. In contrast to other methods, we explicitly take the computation time of the selected features into account. The method is conceptually simple, easy to implement, and allows alternate utility functions, e.g., based on mutual information, as well as additional optimization goals such as the measurement uncertainty of a feature. We verified our method with one synthetic and two real-world data sets from the realm of bulk material sorting. We find that with real-world datasets, the selected feature sets are highly suitable, especially when considering real-time systems. From our analysis it is not clear whether a filter or a wrapper criterion is superior. Which is more suitable depends on the dataset.

As mentioned in section 3, the cost of each feature is assumed to be independent from the selection. In the future, we plan to investigate how dependencies between features can be exploited, for example by modelling these dependencies in a graph structure. Furthermore, we will investigate whether switching to a less accurate, but faster set of features while a system is running can be beneficial to a real-time system under heavy load. In this context, we will also investigate methods to automatically identify which set is most suited for a given system load.

References

[1] H. Akaike. A new look at the statistical model identification. *Automatic Control*, 19(6):716–723, 1974.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] A. Burla, T. Haist, W. Lyda, and W. Osten. Genetic programming applied to automatic algorithm design in multi-scale inspection systems. *Optical Engineering*, 51(6):067001–1, 2012.

[4] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, Jan. 2014.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. In *Evolutionary Computation*, volume 6, pages 182–197, Apr 2002.

[6] M. Ebner. Towards Automated Learning of Object Detectors. In *Applications of Evolutionary Computation*, number 6024 in Lecture Notes in Computer Science, pages 231–240. Springer Berlin Heidelberg, Apr. 2010.

[7] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, pages 545–552, 2004.

[8] M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.

[9] T. M. Hamdani, J.-M. Won, A. M. Alimi, and F. Karray. Multi-objective Feature Selection with NSGA II. In *Adaptive and Natural Computing Algorithms*, Lecture Notes in Computer Science, pages 240–247. Springer Berlin Heidelberg, 2007.

[10] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, Jan. 1968.

[11] K. Iswandy and A. Koenig. Feature selection with acquisition cost for optimizing sensor system design. *Advances in Radio Science*, 4(7):135–141, 2006.

[12] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li. *Applied Linear Statistical Models*. McGraw-Hill/Irwin series. McGraw-Hill Irwin, 2005.

[13] K. Lillywhite, D.-J. Lee, B. Tippetts, and J. Archibald. A feature construction method for general object recognition. *Pattern Recognition*, 46(12):3300–3314, 2013.

[14] B. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Bio-chem Biophy Acta*, 405(2):442–451, Oct. 1975.

[15] M. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In *2013 12th International Conference on Document Analysis and Recognition*, volume 2, pages 666–666. IEEE Computer Society, 2003.

[16] P. M. Narendra and K. Fukunaga. A Branch and Bound Algorithm for Feature Subset Selection. *Computers*, C-26(9):917–922, Sept 1977.

[17] P. Paclík, R. P. W. Duin, G. M. P. v. Kempen, and R. Kohlus. On Feature Selection with Measurement Cost and Grouped Features. In *Structural, Syntactic, and Statistical Pattern Recognition*, number 2396 in Lecture Notes in Computer Science, pages 461–469. Springer Berlin Heidelberg, Aug. 2002.

[18] J. Plasberg and W. Kleijn. Feature Selection Under a Complexity Constraint. *Multimedia*, 11(3):565–571, Apr. 2009.

[19] M. Richter and J. Beyerer. Optical filter selection for automatic visual inspection. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 123–128. IEEE, 2014.

[20] Saroj and Jyoti. Multi-objective genetic algorithm approach to feature subset optimization. In *Advance Computing Conference (IACC)*, pages 544–548, Feb 2014.

[21] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern recognition letters*, 10(5):335–347, 1989.

[22] S. D. Stearns. On selecting features for pattern classifiers. In *International Conference on Pattern Recognition*, pages 71–75, Coronado, CA, 1976.

[23] U. Tekguc, H. Soyel, and H. Demirel. Feature selection for person-independent 3D facial expression recognition using NSGA-II. In *Computer and Information Sciences*, pages 35–38. IEEE, 2009.

[24] B. Xue, M. Zhang, and W. Browne. Particle swarm optimization for feature selection in classification: A multi-objective approach. *Cybernetics*, 43(6):1656–1671, Dec 2013.

[25] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.