Distributed Feature Selection for Multi-Class Classification Using ADMM

Daniel Jung

Jung, D., (2021), Distributed Feature Selection for Multi-Class Classification Using ADMM, *IEEE CONTROL SYSTEMS LETTERS*, 5(3), 821-826. https://doi.org/10.1109/LCSYS.2020.3006428

Distributed Feature Selection for Multi-Class Classification Using ADMM

Daniel Jung

***Abstract*—Feature selection is an important task in data-driven control applications to identify relevant features and remove non-informative ones, for example residual selection for fault diagnosis. For multi-class data, the objective is to find a minimal set of features that can distinguish data from all different classes. A distributed feature selection algorithm is derived using convex optimization and the Alternating Direction Method of Multipliers. The distributed algorithm scales well with increasing number of classes by utilizing parallel computations. Two case studies are used to evaluate the developed feature selection algorithm: fault classification of an internal combustion engine and the MNIST data set to illustrate a larger multi-class classification problem.**

***Index Terms*—Pattern recognition and classification, Optimization algorithms, Fault diagnosis.**

I. Introduction

FEATURE selection is an important part of data-driven classification to identify the most important features to distinguish data from different sets of classes [5], [11], [20]. Examples of applications of feature selection for multi-class classification can be found in medicine [19], [16], [26], text classification [23], and system monitoring [6], [14]. Reducing the number of features helps improve robustness and interpretability of data-driven models by removing non-informative features and avoid overfit [21].

One application of feature selection is fault diagnosis where the objective is to identify a (minimal) subset of the measured signals or features to track the system health. In model-based diagnosis, residual generators are computed based on a model derived from physical insight about the system to detect inconsistencies between sensor data and model predictions [2]. A set of residuals are designed to monitor different parts of the system by generating various fault patterns that can be used to detect faults and isolate their root cause [25]. By designing residuals using different parts of the system model, it is possible to decouple the effect of certain fault classes on the residual output [14].

The number of residual candidates grows exponentially with the number of available sensors [17]. Computer-aided tools, such as [9], can automatically generate residuals from a given system model which makes it possible to explore and evaluate a larger set of residual candidates. Due to model inaccuracies and sensor noise, all residual candidates are not equally good and feature selection is crucial to design a diagnosis system that achieves satisfactory fault diagnosis performance [14].

*This work was not supported by any organization

D. Jung is with the Department of Electrical Engineering, Linköping University, Linköping, Sweden daniel.jung@liu.se

A feature selection algorithm using convex optimization for multi-class classification was proposed in [14] to solve the residual selection problem. The proposed algorithm formulates the multi-class feature selection problem as a single convex optimization problem to find a minimal set of features to distinguish data from the different classes. Collecting representative faulty data is a complicated and expensive process since faults are rare events. One of the main motivations for the proposed feature selection algorithm in [14] is to assure that the selected residuals are able to distinguish between the fault classes, even though available training data from different fault realizations are limited. This is done by taking the information about fault sensitivities of different model-based residuals into consideration when formulating the optimization problem.

If the candidate set is large, or if training data contain many classes, solving the optimization problem can be computationally intractable. With respect to the previous work in [14], a distributed feature selection algorithm is derived here using the *Alternating Direction Method of Multipliers* (ADMM) [3]. The proposed algorithm scales better with increasing number of feature candidates and data classes by reducing the optimization problem into a set of simpler subproblems and distributing the computations over multiple cpus.

ADMM has also been used for multi-class feature selection in, for example, [7] and [24]. In [1], feature selection for intrusion detection is formulated using ADMM and a support-vector classification-regression approach. With respect to mentioned works, the proposed feature selection algorithm models separation between different classes using the logistic regression model and can include requirements on data separation between classes in the optimization problem.

A. Feature Selection for Multi-Class Classification

Consider a multi-class feature selection problem with n feature candidates and m data classes. In [14], the feature selection problem is formulated as the following convex optimization problem

$$\min_{\substack{x_0,x_i,\chi_i \\ i=1,2,\ldots,q}} \quad \|x_0\|_1 \tag{1a}$$

$$\text{s. t.} \quad \Phi_i(x_i,\chi_i) \geq c_i \tag{1b}$$

$$-x_0 \preceq x_i \preceq x_0 \tag{1c}$$

$$\forall i = 1,2,\ldots,q \tag{1d}$$

where $x_0, x_i \in \mathbb{R}^n$, $\chi_i \in \mathbb{R}$, and $\preceq$ represents element-wise inequality. A non-zero element in the solution x_0^* to (1) means that the corresponding feature is selected. Pair-wise separation between the m classes gives $q = m(m-1)/2$ constraints.

Each constraint (1b) represents a performance requirement i, distinguishing two specific fault classes from each other, that is quantified as

$$\Phi_i(x_i, \chi_i) = -\sum_{t=1}^{N} \log\left(1 + \exp\left(-\psi_t \bar{r}_t \bar{x}_i\right)\right) \tag{2}$$

which is the maximum likelihood expression of a logistic regression model [12]. The notation $\bar{x}_i = \begin{pmatrix} x_i^{\mathrm{T}} & \chi_i \end{pmatrix}^{\mathrm{T}}$ is used for a more compact formulation and $\bar{r}_t = \begin{pmatrix} r_1 & r_2 & \dots & r_n & 1 \end{pmatrix}$ is a row vector and denotes the tth sample of all features where the vector is augmented with a constant one to be multiplied with χ_i. Each parameter c_i in (1b) represents a lower bound on (2) and tuned such that satisfactory separation between each pair of classes is achieved. A large value of c_i corresponds to a bigger separation between the classes, according to (2), and will, in general, result in a larger solution set, i.e. more non-zero elements in x_0. The training data set $R_i = \begin{pmatrix} \bar{r}_1^{\mathrm{T}} & \bar{r}_2^{\mathrm{T}} & \dots & \bar{r}_N^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}}$ in (2) used for each constraint i contains samples from the two classes, considered in that constraint, to be distinguished from each other and $\psi_t \in \{-1, 1\}$ is used to label the two classes.

B. Residual Selection Using Fault Sensitivity Information

For model-based residual selection, the optimization problem (1) can take fault sensitivity information about the different residuals into consideration by modifying the constraints (1b) and (1c) accordingly [14]. Let $\bar{r}_t \in \mathbb{R}^{n_i+1}$ in each constraint i in (2) only include residuals with appropriate fault sensitivities. Let $x_i \in \mathbb{R}^{n_i}$ where $n_i \leq n$. The bounds (1c) are replaced with $-x_0^i \preceq x_i \preceq x_0^i$ where x_0^i denotes the subset of elements in x_0 that corresponds to the elements in vector x_i [14]. In the case studies considered in this work, there are no information about fault sensitivities available. Thus, to keep the notation simple, the proposed ADMM algorithm will be derived based on the problem formulation (1).

II. Problem Statement

The dimension of the optimization problem (1) is proportional to $\mathcal{O}(nm^2)$ which can be computationally demanding if the number of features or fault classes is large. The structure of the optimization problem (1) is sparse, since each set of variables $\{x_i, \chi_i\}$ is only included in one set of constraints. This makes it possible to partition the optimization problem into several smaller subproblems and solve them iteratively using distributed optimization methods.

Here, a distributed feature selection algorithm using ADMM is proposed. The optimization problem is based on (1) and finds a sparse solution using an L1 penalty cost function (1a). To illustrate the proposed method, two case studies are evaluated. First, feature selection is performed for fault diagnosis using real residual data collected from an internal combustion engine during different fault scenarios [13]. To analyze the performance on larger problems with many feature candidates and classes of data, the MNIST data set is used where the objective is to select a subset of pixels to distinguish between different written numbers [18].

III. Distributed Feature Selection using ADMM

In (1), an L1 penalty on x_0 is used as a cost function to find a minimal solution set. If a feature is used to fulfill one constraint (1c), the same feature can be used for *free* to fulfill other constraints as well as long as the element in $|x_i|$ is smaller than the corresponding element in x_0. If $x_i^* = \arg_{x_i} \max_{x_i, \chi_i} \Phi_i(x_i, \chi_i)$ is within the boundary given by (1c), there is a risk of overfit of (2) if many features are strongly correlated and training data are limited [15]. A reformulation of the optimization problem (1), which also includes L2 penalties for each x_i, is given by

$$\begin{aligned} \min_{\substack{x_0, x_i, \chi_i \\ i=1,2,\dots,q}} \quad & \|x_0\|_1 + \frac{\lambda}{2}\sum_{i=1}^{q} \|x_i\|_2^2 + \\ & + \sum_{i=1}^{q} \Pi_{\mathcal{C}_i}(x_i, \chi_i) + \sum_{i=1}^{q} \mathcal{I}(\zeta_i) + \sum_{i=1}^{q} \mathcal{I}(\xi_i) \\ \text{s. t.} \quad & -x_i - x_0 = \zeta_i \\ & x_i - x_0 = \xi_i \qquad\qquad \forall i = 1, 2, \dots, q \end{aligned} \tag{3}$$

where λ is a tuning parameter, the vectors ζ_i and ξ_i are slack variables to reformulate the inequality constraints (1c) as equality constraints, and

$$\mathcal{I}(y) = \begin{cases} 0 & \text{if } y \leq 0 \\ +\infty & \text{otherwise} \end{cases} \tag{4}$$

$$\Pi_{\mathcal{C}_i}(x_i, \chi_i) = \begin{cases} 0 & \text{if } \Phi_i(x_i, \chi_i) \geq c_i \\ +\infty & \text{otherwise} \end{cases} \tag{5}$$

are indicator functions.

A. Distributed Optimization Using ADMM

The distributed optimization using ADMM is formulated by defining the Augmented Lagrangian [3]. The scaled dual form of the Augmented Lagrangian of (3) can be written as

$$\begin{aligned} L = & \|x_0\|_1 + \frac{\lambda}{2}\sum_{i=1}^{q} \|x_i\|_2^2 + \\ & + \sum_{i=1}^{q} \left(\Pi_{\mathcal{C}_i}(x_i, \chi_i) + \mathcal{I}(\zeta_i) + \mathcal{I}(\xi_i)\right) + \\ & + \frac{\rho}{2}\sum_{i=1}^{q} \left(\|-x_i + x_0 + \xi_i + \gamma_i\|_2^2 - \|\gamma_i\|_2^2\right) + \\ & + \frac{\rho}{2}\sum_{i=1}^{q} \left(\|x_i + x_0 + \zeta_i + \mu_i\|_2^2 - \|\mu_i\|_2^2\right) \end{aligned} \tag{6}$$

where γ_i and μ_i, $i = 1, 2, \dots, q$, are Lagrangian variables and ρ is a tuning parameter representing the step length of the ADMM algorithm. Minimizing the Augmented Lagrangian is equivalent to solving (3) in the sense that the solution that minimizes (6) also minimizes (3).

The iterative variable updating strategy in ADMM is given by differentiating the Augmented Lagrangian (6) with respect to each variable and then solve them iteratively, one at the time, based on already updated variables. One advantage of this approach is that when some variables can be computed independently of the others, a sequence of simpler problems

can be formulated to solve the original optimization problem that are much faster to evaluate.

1) Updating Rules: The updating rule for x_0 is derived by differentiating (6) with respect to x_0 and solve when the derivative is equal to zero. Let $\text{sign}(\cdot)$ denote element-wise sign. Then, the partial derivative $\frac{\partial L}{\partial x_0}$ can be written as

$$\frac{\partial L}{\partial x_0} = \text{sign}(x_0) + \rho \sum_{i=1}^{q} (2x_0 + \xi_i + \gamma_i + \zeta_i + \mu_i) = 0$$

The vector x_0 can be computed using element-wise max as

$$\begin{aligned} x_0 = &\frac{1}{2}\left[\max\left(0, -\bar{\zeta} - \bar{\xi} - \bar{\mu} - \bar{\gamma} - \frac{1}{\rho q}\right) - \right. \\ &\left. \max\left(0, \bar{\zeta} + \bar{\xi} + \bar{\mu} + \bar{\gamma} - \frac{1}{\rho q}\right)\right] \end{aligned} \tag{7}$$

where $\bar{y} = \frac{1}{q}\sum_{i=1}^{q} y_i$ denotes the arithmetic mean.

Differentiating (6) with respect to the slack variables ξ_i and ζ_i for $i = 1, \ldots, q$ gives the updating rules

$$\xi_i = \min\left(0, x_i - x_0 - \gamma_i\right) \quad \zeta_i = \min\left(0, -x_i - x_0 - \mu_i\right) \tag{8}$$

where the minimization is performed element-wise.

An explicit updating rule for each vector x_i and χ_i cannot be derived. Instead, they can be computed by solving the following constrained optimization problem

$$\begin{aligned} \min_{x_i, \chi_i} \quad & \frac{\lambda}{2}\|x_i\|_2^2 + \frac{\rho}{2}\|-x_i + x_0 + \xi_i + \gamma_i\|_2^2 + \\ & + \frac{\rho}{2}\|x_i + x_0 + \zeta_i + \mu_i\|_2^2 \\ \text{s. t.} \quad & \Phi(x_i, \chi_i) \geq c_i \end{aligned} \tag{9}$$

where the indicator function $\Pi_{\mathcal{C}_i}(x_i, \chi_i)$ is implemented as a constraint.

To implement an efficient solver for (9), using e.g. Newton-type methods, analytic expressions for the gradient g_i and Hessian H_i of $\Phi_i(x_i, \chi_i)$ are given by $g_i = -R_i^{\text{T}}(p - \psi)$ and $H_i = -R_i^{\text{T}} W R_i$ where p is a column vector and each element t is given by $p(\bar{r}_t; \bar{x}_i) = \left(1 + \exp\left(-\psi_t \bar{r}_t \bar{x}_i\right)\right)^{-1}$ and W is a diagonal matrix where the diagonal element at position (t, t) is given by $p(\bar{r}_t; \bar{x}_i)(1 - p(\bar{r}_t; \bar{x}_i))$ [12]. Also, the gradient of the cost function in (9) is given by $\begin{pmatrix} (\lambda + 2\rho) x_i + \rho\left(-\xi_i + \zeta_i - \gamma_i + \mu_i\right) \\ 0 \end{pmatrix}$ and the Hessian is $\begin{pmatrix} (\lambda + 2\rho) I & 0_n \\ 0_n^{\text{T}} & 0 \end{pmatrix}$ where $I \in \mathbb{R}^{n \times n}$ is an identity matrix and $0_n \in \mathbb{R}^n$ is a zero column vector.

The Lagrangian variables are updated as follows [3]:

$$\mu_i := \mu_i + x_i + x_0 + \zeta_i \quad \gamma_i := \gamma_i - x_i + x_0 + \xi_i \tag{10}$$

The steps updating the slack variables (8), optimizing x_i and χ_i in (9), and updating the Lagrangian variables (10) only depend on x_0 and variables associated to constraint i, i.e. no variables from other constraints $j \neq i$. These steps can be solved in parallel for each constraint, independent of each other, in the ADMM algorithm.

2) Algorithm Summary: Almost all steps of the ADMM algorithm can be solved explicitly except (9). Thus, if there are (at least) as many parallel cores available as the number of constraints, the computational complexity of each iteration is dominated by the maximum time it takes to solve (9) for any constraint $i = 1, 2, \ldots q$, e.g. using interior-point methods which will have polynomial-time complexity [15]. The steps of the proposed feature selection algorithm are summarized in Algorithm 1 where **parfor** (parallel for-loop) is used to emphasize that each iteration of the for-loop can be run in parallel. The superscript k denotes iteration and shows when each updated variable is used to compute the next one.

Algorithm 1 A summary of the distributed convex feature selection algorithm using ADMM

```
# Initiate variables
k=0, x_0=0, x_i=0, x_i^0=0, γ_i=0, μ_i=0, ζ_i=0, ξ_i=0   ∀i
repeat
  x_0^{k+1} := 1/2 [max(0, -ζ̄^k - ξ̄^k - μ̄^k - γ̄^k - 1/(ρq)) -
  - max(0, ζ̄^k + ξ̄^k + μ̄^k + γ̄^k - 1/(ρq))]
  # Parallel for loop
  parfor i = 1 to q do
    # Update slack variables
    ξ_i^{k+1} := min(0, x_i^k - x_0^{k+1} - γ_i^k)
    ζ_i^{k+1} := min(0, -x_i^k - x_0^{k+1} - μ_i^k)
    # Optimization
    x_i^{k+1}, χ_i^{k+1} := solve (9) using x_0^{k+1}, x_i^k, χ_i^k, γ_i^k, μ_i^k,
    ξ_i^{k+1}, and ζ_i^{k+1}.
    # Update dual variables
    μ_i^{k+1} := μ_i^k + x_i^{k+1} + x_0^{k+1} + ζ_i^{k+1}
    γ_i^{k+1} := γ_i^k - x_i^{k+1} + x_0^{k+1} + ξ_i^{k+1}
  end parfor
  k := k + 1
until convergence
return x_0^k, x_i^k, χ_i^k   ∀i
```

IV. Evaluation

The main objective in the two case studies is to evaluate Algorithm 1. Both the reference algorithm (1) and Algorithm 1 have been implemented in Matlab for both case studies. The requirement parameters are here selected as $c_i = \max_{x_i, \chi_i} \Phi_i(x_i, \chi_i) - b_i$ where the constant $b_i > 0$ is experimentally tuned to give satisfactory performance in each constraint i. To numerically identify non-zero elements in x_0, the solution set contains the features corresponding to vector elements exceeding a selected threshold of 0.01.

A. Residual Selection for Engine Fault Diagnosis

The first case study is the same residual selection problem for fault classification of an internal combustion engine as considered in [13]. The purpose is to evaluate the proposed method with respect to the original method in [14].

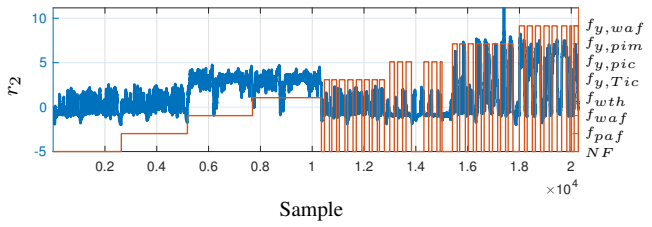

Fig. 1. An example of residual data from all different fault classes.

1) System Description Summary: Sensor data have been collected from the engine test bed during different transient operating cycles including nominal system behavior and seven different single-fault scenarios: air filter clogging; leakages at the air filter and at the throttle; faults in sensors measuring the temperature and pressure at the intercooler, pressure in the intake manifold, and the air flow through the air filter. The faults are injected during operation either into the engine control unit, for example sensor faults, or by physically manipulating the engine, for example air leakages.

The feature candidate set is a set of 42 residual generators $\{r_1, \ldots, r_{42}\}$ that has been generated from a mathematical model of the engine, using the Fault Diagnosis Toolbox in Matlab [9]. The set of residual candidates is the same as used in [8] and an example of residual data is shown in Fig. 1 also showing the class label for each sample[1]. The residuals are evaluated using the collected data from the different faults where 10% of the residual output data are used as training data [13], i.e. 2024 samples in total for all classes. All residuals have been normalized to have zero mean and variance one for the nominal class (No Fault - NF).

2) Results: The eight different data classes (nominal and seven faults) result in 28 performance constraints to evaluate pairwise separation between all classes. The convergence rate of the proposed algorithm is evaluated by comparing the results to the solution to (1) when $\lambda = 0$, i.e. no L2 penalties [13]. Algorithm 1 is evaluated with step length $\rho = 0.1$ and the error tolerance level when solving (9) is selected as 10^{-4} during the first 400 iterations and 10^{-6} during the following iterations. As proposed in [3], computation time can be improved by selecting a higher error tolerance when solving (9) during the first iterations of the ADMM algorithm and then lower it to achieve better numerical accuracy when the solution is closer to the global optimum.

The solution x_0 per iteration is shown in Fig. 2 where the dashed lines represent the global solution computed using (1). The ADMM algorithm converges to the global solution given (1) within 450 iterations but the solution set, i.e. the resulting non-zero elements in x_0, can be identified already after 100 iterations. The computation time is presented in Fig. 3 showing that computation time is higher in the initial iterations but decreases significantly with increasing number of iterations. This is expected as the interior-point method used to solve (9) in Algorithm 1 should converge faster when the solution x_0 is

[1]Residual data are available in the Fault Diagnosis Toolbox [9] that can be downloaded from https://faultdiagnosistoolbox.github.io. The selected residual subset used in this work is described in [13].

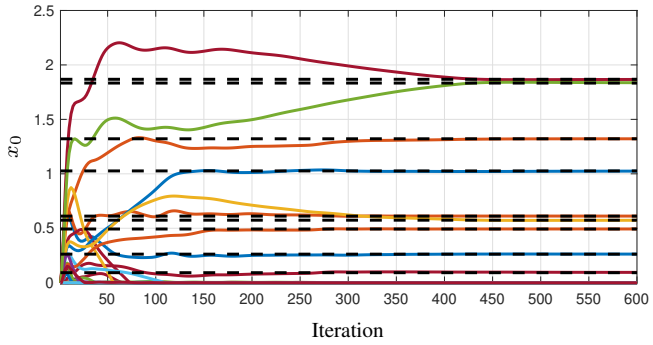

Fig. 2. The solution when applying the ADMM algorithm to the engine case study. Results show that the solution converges to the global solution and the solution set (features corresponding to the non-zero elements in x_0) can be identified already after 100 iterations.

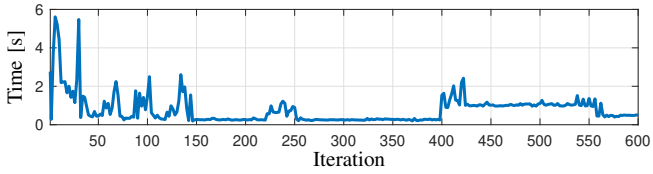

Fig. 3. The computation time per iteration of the ADMM algorithm applied to the engine case study. The tolerance level when solving (9) is reduced from 10^{-4} to 10^{-6} after 400 iterations resulting in longer computation time.

stabilizing and the solution from the previous iteration is used as a warm start. After iteration 400, the error tolerance level is changed resulting in a higher computation time.

There is no rule how to select ρ in the general case. However, some results are derived for specific classes of problems, see e.g. [10]. Fig. 4 compares the convergence rate for different choices of ρ by showing the square root error with respect to the global optimum. Careful tuning of ρ is important since both too small and too large step sizes require longer time to converge which is consistent with observations in [10]. In this case, the fastest convergence rate is achieved for $\rho = 0.1$. The original optimization problem (1) is solved in less than five seconds on a standard laptop, while the ADMM approach takes longer time. The advantage of the ADMM formulation becomes clear for larger problems as illustrated in the next case study.

B. MNIST Data

To analyze the computational aspects when the feature selection problem grows, i.e. a larger number of candidate features and data classes, the MNIST image classification data

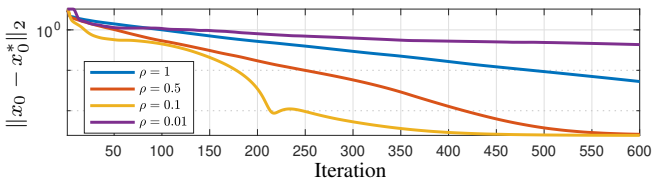

Fig. 4. The convergence rate for different step sizes ρ in the engine case study. Fastest convergence rate is achieved for $\rho = 0.1$.

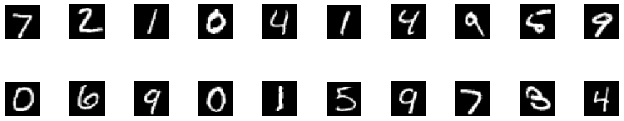

Fig. 5. An example of image samples in the MNIST data set.

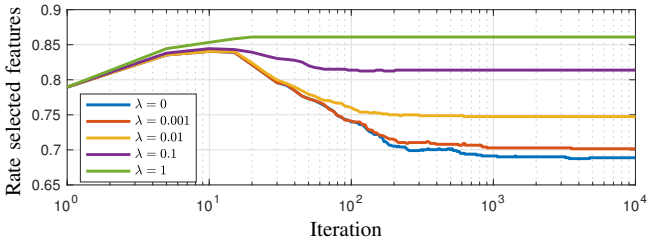

Fig. 6. The size of the solution set in the MNIST data case study for different values of λ at different iterations.

set is used [18]. Training data consists of approximately 60 000 images of digits from 0 to 9 where each picture consists of $28 \times 28 = 784$ pixels with binary values, see Fig. 5. In this analysis, each pixel is considered a candidate feature and the ten different numbers result in 45 constraints. This means that the feature candidate set is almost 19 times larger compared to the engine case study and includes 50% more constraints.

The feature selection problem (1) could not be solved for the MNIST data set on a standard laptop with dual cores because of memory limitations. Algorithm 1 is evaluated at the Swedish National Super Computer Centre (NSC) using 46 parallel cores. Based on the results from the engine diagnosis case study, the step size $\rho = 0.1$ is selected in this analysis.

Compared to the engine residual data case study, the distributed feature selection of the MNIST data requires more iterations before convergence which is expected since the problem is significantly larger. When analyzing the size of the solution set in Fig. 6, the ADMM algorithm filters out most of the non-important features already after around 1000 iterations and converges to selecting around 69% of the pixels, i.e. identifying 31% of the features as non-important.

The effect of the L2 penalty is evaluated by increasing λ from zero to one. The number of selected features increases up to 86% for $\lambda = 1$. The computation time decreases significantly when $\lambda > 0$ as shown in Fig. 7. A tolerance error level of 10^{-4} is used during the first 1000 iterations which is then changed to 10^{-6} in the following iterations resulting in a longer iteration time. Notice that choosing $\lambda = 0.001$ significantly decreases computation time with respect to $\lambda = 0$ and does not increase the size of the solution set significantly, see Fig. 6.

1) Feature Selection Using Random Forest: For comparison, feature importance is evaluated using the Random Forest (RF) algorithm [4]. Random Forests can quantify variable importance in multi-class classification problems using VIMP (Variable IMPortance) which measures the performance loss when a variable is randomized. Fig. 8 compares the solutions x_0 of Algorithm 1 for $\lambda = 0$ and $\lambda = 0.1$ with VIMP. The values in solution x_0 that exceed 50 have been saturated to improve visualization. The results show that both methods

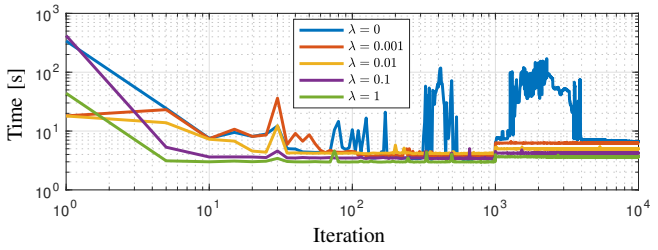

Fig. 7. The computational time per iteration for different choices of λ in the MNIST data case study. Using a $\lambda > 0$ reduces computation time at the cost of a larger solution set.

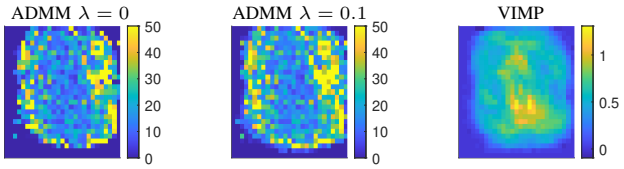

Fig. 8. A comparison between Algorithm 1 for $\lambda = 0$ and $\lambda = 0.1$ and Random Forest using VIMP. The colorbars show variable importance of each pixel measured using x_0 from Algorithm 1 in the left and middle plots and VIMP in the right plot. Note that values of x_0 that exceed 50 have been saturated to improve visualization.

identifies more or less the same pixels as non-important. However, RF values the pixels in the center of the picture as most important while Algorithm 1 identifies the surrounding pixels as important.

To compare the feature selection algorithms, a set of RF classifiers with 1000 trees are trained based on the selected features in Fig. 8 and their performance are evaluated based on their out-of-bag (OOB) error rate. Algorithm 1 with $\lambda = 0$ and $\lambda = 0.1$ select 65.8% and 81.3% features, respectively, while 82.3% features are selected using VIMP (values greater than 10^{-3}). The OOB error rates are almost identical to training a RF using all pixels as features, where the error rates for the evaluated feature sets are all in the interval $[2.86\%, 2.94\%]$. The results show that Algorithm 1 finds a feature set that is 17% smaller compared to VIMP without affecting classification performance.

C. A Heuristic Stopping Criteria

If the solution of the proposed feature selection algorithm is used as an input to train another classifier, it is not necessary that the algorithm has converged as long as the solution set is feasible. Fig. 6 shows that the size of the solution set decreases mostly during the initial iterations and there were similar observations in the engine fault diagnosis case study. Therefore, a heuristic solution to reduce computation time is to make an early stop when a feasible solution size is found and the solution set is stabilizing. The feasibility of the solution can be evaluated by analyzing the Lagrangian variables μ_i and γ_i. A feasible solution should have all μ_i and γ_i to be numerically zero.

A heuristic stopping criteria is to select a threshold for μ_i and γ_i when a solution is considered numerically feasible. Fig. 9 shows $\|\mu\|_2 = \sqrt{\sum_{i=1}^{p} \mu_i^2}$ and $\|\gamma\|_2 = \sqrt{\sum_{i=1}^{p} \gamma_i^2}$ for the feature selection of the MNIST data set per iteration

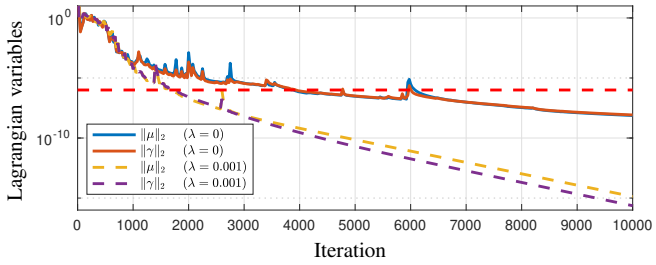

Fig. 9. Lagrangian variables as functions of iteration from the MNIST data case study. The dashed line represents a threshold of 10^{-6} that can be used as a heuristic stopping criteria. The Lagrangian variables decrease faster when adding an L2 penalty.

when $\lambda = 0$ and $\lambda = 0.001$. The dashed line represents a tolerance threshold selected at level 10^{-6}. The Lagrangian variables decrease faster when $\lambda = 0.001$ and goes below the threshold after approximately 1500 iterations compared to 4000 iterations for $\lambda = 0$. Results show a faster decrease for larger values of λ which would result in an earlier stop. There are a few spikes in the curves in Fig. 9 making the Lagrangian variables exceed the selected threshold, e.g. around 6000 iterations. An analysis of the solution x_0 as a function of iteration shows that the spikes when the Lagrangian variables are small only appear to relate to small variations in elements of x_0 that are close to zero. Fig. 2 shows that the number of features have already stabilized when the Lagrangian variables pass the threshold in Fig. 9 and can be chosen as acceptable solution sets. Similar observations are found when looking at the engine fault diagnosis data set where the variables μ_i and γ_i go below the tolerance threshold at around iteration 500 when the solution sets in Fig. 2 have already stabilized.

The analysis indicates that the computation time can be improved, without having to significantly compromise the solution size, e.g. by tuning error tolerance levels, choosing a small positive λ, and apply early stopping by analyzing the magnitudes of the Lagrangian variables.

V. Conclusion

Multi-class feature selection is an important task in many data-driven control applications, such as fault diagnosis. The proposed feature selection algorithm is formulated as a distributed convex optimization problem using ADMM meaning that it will converge to the global optimum. Two real data case studies show that the proposed feature selection problem can solve larger problems compared to a non-distributed formulation. It is also shown that applying varying error tolerance levels, L2 regularization, and heuristic rules for early stopping, can significantly improve computational time.

As part of future work, it is relevant to analyze the theoretical convergence rate of the proposed algorithm and finding an optimal tuning strategy of the step length ρ. It could also be possible to improve the convergence rate by modifying Algorithm 1 using, for example, proximal operators [22].

Acknowledgement

The data analysis has been conducted at the Swedish National Super Computer Center (NSC).

References

[1] S. Bamakan, H. Wang, and Y. Shi. Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem. *Knowledge-Based Systems*, 126:113–126, 2017.

[2] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder. *Diagnosis and fault-tolerant control*, volume 2. Springer, 2006.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] J. Cai, J. Luo, S. Wang, and S. Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[6] M. Cerrada, R. Sánchez, F. Pacheco, D. Cabrera, G. Zurita, and C. Li. Hierarchical feature selection based on relative dependency for gear fault diagnosis. *Applied Intelligence*, 44(3):687–703, 2016.

[7] L. Chen, J. Tang, and B. Li. Embedded supervised feature selection for multi-class data. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 516–524, 2017.

[8] E. Frisk and M. Krysander. Residual selection for consistency based diagnosis using machine learning models. In *IFAC SafeProcess*, Warszaw, Poland, August 2018.

[9] E. Frisk, M. Krysander, and D. Jung. A toolbox for analysis and design of model based diagnosis systems for large scale models. In *IFAC World Congress*, Toulouse, France, 2017.

[10] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[11] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3:1157–1182, 2003.

[12] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[13] D. Jung. Engine fault diagnosis combining model-based residuals and data-driven classifiers. In *Proc. of IFAC International Symposium on Advances in Automotive Control*, Orléans, France, June 2019.

[14] D. Jung and E. Frisk. Residual selection for fault detection and isolation using convex optimization. *Automatica*, 97:143–149, 2018.

[15] K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine learning research*, 8(Jul):1519–1555, 2007.

[16] Y. Kong, Y. Deng, and Q. Dai. Discriminative clustering and feature selection for brain mri segmentation. *IEEE Signal Processing Letters*, 22(5):573–577, 2014.

[17] M. Krysander, J. Åslund, and M. Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1):197–206, 2007.

[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.

[20] J. Miao and L. Niu. A survey on feature selection. *Procedia Computer Science*, 91:919–926, 2016.

[21] A. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[22] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[23] B. Tang, S. Kay, and H. He. Toward optimal feature selection in naive bayes for text categorization. *IEEE transactions on knowledge and data engineering*, 28(9):2508–2521, 2016.

[24] F. Tang, L. Adam, and B. Si. Group feature selection with multiclass support vector machine. *Neurocomputing*, 317:42–49, 2018.

[25] L. Travé-Massuyès. Bridging control and artificial intelligence theories for diagnosis: A survey. *Engineering Applications of Artificial Intelligence*, 27:1–16, 2014.

[26] X. Zhu, H. Suk, S. Lee, and D. Shen. Canonical feature selection for joint regression and multi-class identification in alzheimer's disease diagnosis. *Brain imaging and behavior*, 10(3):818–828, 2016.