Differentiable Feature Selection by Discrete Relaxation

Rishit Sheth **Nicolo Fusi**

Microsoft Research New England

Abstract

In this paper, we introduce Differentiable Feature Selection, a gradient-based search algorithm for feature selection. Our approach extends a recent result on the estimation of learnability in the sublinear data regime by showing that the calculation can be performed iteratively (*i.e.,* in mini-batches) and in linear time and space with respect to both the number of features D *and* the sample size N. This, along with a discrete-to-continuous relaxation of the search domain, allows for an efficient, gradient-based search algorithm among feature subsets for very large datasets. Our algorithm utilizes higher-order correlations between features and targets for both the $N > D$ and $N < D$ regimes, as opposed to approaches that do not consider such correlations and/or only consider one regime. We provide experimental demonstration of the algorithm in small and large sample- and feature-size settings.

1 Introduction

We consider the problem of feature selection in supervised learning tasks. Feature selection remains a crucial step in machine learning pipelines and continues to see active research (e.g., Aghazadeh et al. (2018); Abid et al. (2019)). Earlier application of feature selection methods, such as those in computational biology, considered settings where the number of features available was large and the sample size was relatively small. The goal was to choose a small subset of features with good explanatory power and doing so required *statistically efficient* feature selection methods. More recent applications of feature selection methods (*e.g.,* in natural language processing) are characterized both by large sample sizes and large feature spaces, making *computational efficiency* another key requirement.

To address these challenges, several feature selection methods have been proposed over the years. These methods can be broadly categorized as either wrapper or filter methods. Wrapper methods utilize a prescribed model to select features by training and evaluating the model on different feature subsets (Kohavi and John, 1997). Filter methods utilize a computationally cheap evaluation criterion to rank features individually or in certain combinations (Yu and Liu, 2003). In addition to hybrid combinations of these basic approaches, embedded feature selection methods that perform joint model selection and training have also been proposed. An example of such hybrid methods is the lasso (Tibshirani, 1996). Generally, wrapper methods give better results since they directly evaluate the final supervised learning task. However, they are also the most computationally demanding, since they require training and evaluating a model per proposed feature subset. Filter methods are considered computationally cheaper alternatives with better scaling, but are generally based on heuristic objectives.

In this paper, our goal is to develop a feature selection method that is statistically efficient (*i.e.,* identifies a high-quality subset of features for a linear model in $N < D$ settings), and computationally efficient (*i.e.,* scales to millions of samples and features). Ideally, the method should consist of primitive operations that can be accelerated using GPUs.

We build such a method starting from a recent result by Kong and Valiant (2018) in which the authors present an estimator for the accuracy induced by a set of features in a linear model. While this estimator has desirable statistical properties, it can only assess the "quality" of a given set of features and doesn't prescribe a procedure for selecting a specific set of features. In the following, we treat feature selection as a combinatorial optimization problem and employ a discrete relaxation to efficiently traverse the space of possible feature sets. Our algorithm combines the

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

strong statistical properties of the Kong and Valiant (2018) estimator with the ability to search over complex optimization spaces via standard gradient descent. Specifically, the method we propose has the following properties:

- *Low computational complexity.* The complexity scales linearly both with the sample size and the number of features, enabling feature selection even in large datasets.

- *High statistical efficiency.* The estimation accuracy degrades sublinearly with increasing dimension allowing for accurate search over large featurization spaces consisting of millions of features. For example, increasing the number of features by a factor $\gamma > 1$ only requires a factor of $\sqrt{\gamma}$ increase in the number of samples to maintain the same level of accuracy.

- *Efficient utilization of higher-order correlations.* Utilizing higher-order feature correlations to improve the quality of selection only incurs a constant increase in time and space complexity with each additional order.

The final property is described next after introducing the objective used to perform search.

2 Differentiable Feature Selection (DFS)

Our algorithm is based on recent work on the learnability of linear models by Kong and Valiant (2018) which provides high probability bounds on the performance of linear predictors in high-dimensional regression (residual variance) and classification (accuracy). In particular, the bounds for the estimators they introduce have a sublinear dependence on dimension, highlighting the possibility of making well-informed decisions on which feature subset would be most useful for a dataset on a task with relatively little data. We start by giving a brief description of these estimators, moving on to show how they can be modified and utilized for feature selection by means of optimization over a continuous space. For clarity of exposition, we focus on the regression case, but the same ideas and approach hold for classification.

Given inputs $X \in \mathbb{R}^{N\times D}$, targets $y \in \mathbb{R}^N$, a positive integer k, and $\{a_i\}_{i=0}^{k-1}$ with $a_i \in \mathbb{R}$, the residual variance estimate for a subset of features denoted by $s \in \{0,1\}^D$ in the linear regression model of Kong and Valiant (2018) is given by

$$f(s) = \frac{y^\top y}{N} - \sum_{i=0}^{k-1} \frac{a_i}{\binom{N}{i+2}} y^\top \mathrm{triud}(X\mathrm{diag}(s)X^\top)^{i+1} y, \tag{1}$$

where $\mathrm{triud}(\cdot)$ denotes the operation that zeros out the lower triangular portion and diagonal entries of a square matrix. The assumptions for the high probability bound to hold include i.i.d. examples/labels and bounded moments[1].

The parameter k defines the order of the estimator and is related to the degree of a "greater-than-linear" polynomial approximation to $h(x) = x$ as described in Kong and Valiant (2018)[2] Increasing k in (1) results in higher-order correlations between features being taken into account in the residual variance computation thereby increasing the quality of the estimate. By increasing the quality, a potentially better set of features can be located for a linear model in those features. We note that the polynomial coefficients $\{a_i\}$ for any order can be computed offline.

Now, we show that (1) can be efficiently computed:

Lemma 2.1. *The function (1) requires $\mathcal{O}(ND)$ time and $\mathcal{O}(N)$ space.*

The following proposition is useful in proving Lemma 2.1.

Proposition 2.1. *For $z \in \mathbb{R}^N$ and $y \in \mathbb{R}^N$, the operation triud$(zz^\top)y$ requires $\mathcal{O}(N)$ time and space.*

Proof. Expanding the first operation, we have

$$\mathrm{triud}(zz^\top)y = \begin{pmatrix} z_1z_2y_2 + z_1z_3y_3 + \cdots + z_1z_Ny_N \\ z_2z_3y_3 + \cdots + z_2z_Ny_N \\ \vdots \\ z_{N-1}z_Ny_N \\ 0 \end{pmatrix}.$$

Letting $u = z \circ y$,

$$\mathrm{triud}(zz^\top)y = \begin{pmatrix} z_1 \sum_{i=2}^N u_i \\ z_2 \sum_{i=3}^N u_i \\ \vdots \\ z_{N-1}u_N \\ 0 \end{pmatrix},$$

where the elements $(\sum_{i=2}^N u_i, \sum_{i=3}^N u_i, \ldots, u_N, 0)$ can be calculated from u with a (reverse) cumulative sum in $\mathcal{O}(N)$. Since computing u and performing the inner product between z and $(\sum_{i=2}^N u_i, \sum_{i=3}^N u_i, \ldots, u_N, 0)^\top$ are also $\mathcal{O}(N)$, the total cost is linear in N. □

[1] Specifically, the 2nd and 4th order moments for the examples and variance for the noise are assumed to be bounded.

[2] See proposition 8.

Proof of Lemma 2.1. For $X \in \mathbb{R}^{N\times D}$, let $X_{:d}$ for $1 \leq d \leq D$ represent the d-th column of X. Then, note

$$y^\top \text{triud}(X\text{diag}(s)X^\top)^{i+1}y = \tag{2}$$

$$= y^\top \Big(\sum_{d=1}^{D} s_d \text{triud}(X_{:d}X_{:d}^\top)\Big)^{i+1} y \tag{3}$$

$$= y^\top \underbrace{\Big(\sum_{d=1}^{D} s_d G_d\Big)\cdots\Big(\sum_{d=1}^{D} s_d G_d\Big)}_{i \text{ terms}} y, \tag{4}$$

where $G_d \triangleq \text{triud}(X_{:d}X_{:d}^\top)$. From Proposition 2.1, it follows that computing $\sum_{d=1}^{D} s_d G_d y$ requires $\mathcal{O}(ND)$ time and $\mathcal{O}(N)$ space. This operation can be iterated i additional times without increasing the time or space complexity. □

Therefore, (1) can be utilized as an efficiently computable filter criterion. However, given a prescribed number of desired features $1 \leq d \leq D$, the use of (1) for *search* over $s \in \{0,1\}^D$ subject to $\|s\|_0 = d$ is limited since the general problem is NP-hard (Natarajan, 1995). To achieve an approximate, tractable solution, we relax the discrete search domain to a continuous domain via the transformation $s = \sigma(v)$ where $v \in \mathbb{R}^D$ and the "squashing" function $\sigma(\cdot) : \mathbb{R} \mapsto [0,1]$ is applied elementwise. The relaxed, unconstrained optimization is then given by $\arg\min_{v\in\mathbb{R}^D} f(\sigma(v))$, which is amenable to solution by gradient-based search in v. Further, mini-batches of data can be utilized for increased efficiency. In our experiments, we enforce model parsimony with the penalty term $\frac{\lambda}{D}\|s\|_1 = \frac{\lambda}{D}\sum_{d=1}^{D}\sigma(v_d)$ where selection of the regularization parameter $\lambda > 0$ can be accomplished with grid search evaluated on a validation set.

The associated bound of Kong and Valiant (2018) for classification (logistic regression) assumes (i) i.i.d. examples/labels, (ii) zero-mean Gaussian examples, and (iii) bounded spectral norm. We further note that the objective (plus penalty) is non-convex and likely has many local optima. However, as we demonstrate in the experiments, the algorithm is able to identify better feature subsets than competing methods in practical applications, where these conditions may not hold.

3 Related work

The research on feature selection is vast, and we cannot hope to provide a list here with any claim to being comprehensive. Instead, we highlight relevant work with a focus on more recent scalable filter approaches. The interested reader can refer to Guyon and Elisseeff (2003) for a broader overview and Saeys et al. (2007) and Forman (2003) for domain-specific reviews in key application areas.

As mentioned previously, wrapper methods generally produce high quality feature subsets since they optimize directly for the desired supervised learning task. However, here, we seek a general scalable solution that can support arbitrary follow-on tasks. Moreover, in the $N \ll D$ regime, approaches that attempt to train a model risk overfitting or underfitting. In contrast, we are proposing method that produces high-quality estimates of feature subset performance.

Standard filter criteria for classification include the one-way ANOVA test among class means and mutual information (computed per feature dimension). Like DFS, the ANOVA computation is linear in sample size and dimension, but is "greedy" in selection in that features are evaluated in isolation.

The filter algorithm of Yu and Liu (2003, 2004) utilizes a normalized mutual information as correlation measure to first identify target-correleated features, and then, from this set, perform rounds of elimination of redundant features, where redundant is also defined w.r.t. the same correlation measure. Assuming an average case performance of eliminating half of all target-correlated features per round, the algorithm complexity is $\mathcal{O}(ND\log D)$; the worst-case complexity is $\mathcal{O}(ND^2)$.

Concrete and related relaxations developed for latent variable models (Jang et al., 2016; Maddison et al., 2016) offer an avenue for embedded feature selection. Of these, the closest work is that of Abid et al. (2019) which presents an unsupervised approach that selects a prescribed number d of features by learning d (parallel) Concrete distributions over the input features. Training an autoencoding neural network architecture yields a subset of features that are useful for reconstructing the full-dimensional inputs, and, to an extent, supporting follow-on supervised learning tasks. However, a neural network architecture must be specified per problem instance and trained which is a problem that can be generally as difficult as subset selection. Convex relaxations of "ℓ_0"-regularized approaches (e.g., Tibshirani (1996)) relax the objective function rather than the search space. Our particular relaxation is more in line with that of Liu et al. (2018).

Of the recent approaches for streaming feature selection (Sun et al., 2009; Yu et al., 2014), the MISSION algorithm (Aghazadeh et al., 2018) most closely considers the same setting we do. MISSION utilizes a Count-Sketch data structure to store a running, compressed gradient of a generalized linear model which provides scalability to very high dimensional datasets. However, the gradient information is effectively based

on a 1st-order estimate of the residual variance.

4 Experiments

We validate the performance of our method for classification tasks in both small and large datasets. Small datasets allow us to carefully examine the performance of the different algorithms we consider in cases where $N \ll D$, whereas the larger datasets are useful to demonstrate scalability to large sample size and large dimension, in both the $N > D$ and $N < D$ regimes. In cases where the total number of features is on the order of the sample size or greater, statistical accuracy will dominate the selection quality. In cases where the sample size is large, computational efficiency will be a limiting factor for selection algorithms. For small datasets, we compare Differentiable Feature Selection to traditional filter methods. In the large-scale setting, where traditional filter methods would be too computationally expensive, we compare to MISSION (Aghazadeh et al., 2018), a recently-proposed algorithm that can perform efficient feature selection by means of count sketching. The datasets used in these experiments are summarized in Table 1.

The evaluation methodology consisted of first running our method and the competitors on train to yield feature subsets of varying sizes. Then, a logistic regression model is fit per subset with train features (batch-trained for small datasets, SGD-trained for large datasets). Finally, the fitted models are evaluated on test with area under the curve (AUC) as the performance metric. In this way, performance curves of AUC as a function of feature subset size are recorded per dataset.

DFS implementation. Prior to performing the continuous search, we (i) center the data matrix, X, per dimension, and (ii) estimate the largest singular value of the covariance of X and divide the centered data matrix by the square root of this value. Both of these operations are performed[3] in order to meet the requirements of the Kong and Valiant (2018) estimator. Specifically, dividing by the square root of the largest singular value allows a single set of degree-k polynomial coefficients to be used for all input data matrices. The centering and singular value estimation are performed on the entire data matrix except in the cases of *webspam* and *criteo* where it is estimated from a sub-sample of size 10,000. Adam (Kingma and Ba, 2014) with learning rate 10^{-1} (and other parameters set to default) with a gradient clipping value of 1.0 (max 2-norm) was used in all cases. For the small data experiments, the stopping criteria were 1000 maximum iterations or the relative change in objective function dropping below 10^{-5}. In the large-scale experiments, training was performed for one epoch. Mini-batch sizes are described in the relevant sections. The squashing function was $\sigma(2x)$ where $\sigma(\cdot)$ is the sigmoid function (this choice of squashing function is equivalent to $\frac{1}{2}(\tanh(x) + 1)$).

[3]The operations are performed only during the feature selection stage. Subsequent training and evaluation use raw data.

Dataset	D	N_{train}	N_{test}
mnist35	784	11,552	1902
gisette	5000	6000	1000
rcv1	47,236	20,242	677,399
webspam	16,609,143	280,000	70,000
criteo	1,000,000	45,840,617	6,042,135

Table 1: Binary classification datasets.

Software is available at `https://github.com/rsheth80/dfs/`.

Small datasets. The smaller dimensional datasets are:

- *mnist35*. Classification of 3s vs. 5s in the MNIST dataset with random noise added to produce a more challenging scenario. Samples are shown in Figure 1.
- *gisette*. Used in the 2003 NIPS feature selection challenge.

Beyond benchmarking the performance of our method, the goal of this experiment is to provide an intuitive visual representation of the features selected by DFS. We first discuss performance on the *mnist35* dataset.

Figure 2 compares the performance of our method with two filter-based methods implemented in scikit-learn. One computes ANOVA F-values between features and targets (SKF) and the other computes mutual information (SKMI). In each subplot, DFS performance is given by the y-axis, and competitor performance is given by the x-axis. As described in the experimental methodology, each point represents the output of feature selection at a given feature subset size denoted by color. Generally, better performance is achieved when increasing the desired number of features (however, as demonstrated in the subplots for *gisette*, this trend does not hold uniformly across datasets).

Our method significantly outperforms both baselines across a range of feature subset sizes. As a whole, the improvement given by DFS is statistically significant at a significance level of 0.01 with a p-value $< 10^{-4}$, based on a paired t-test between AUCs obtained by each baseline and AUCs obtained by our method.

Figure 1: *mnist35* input examples.

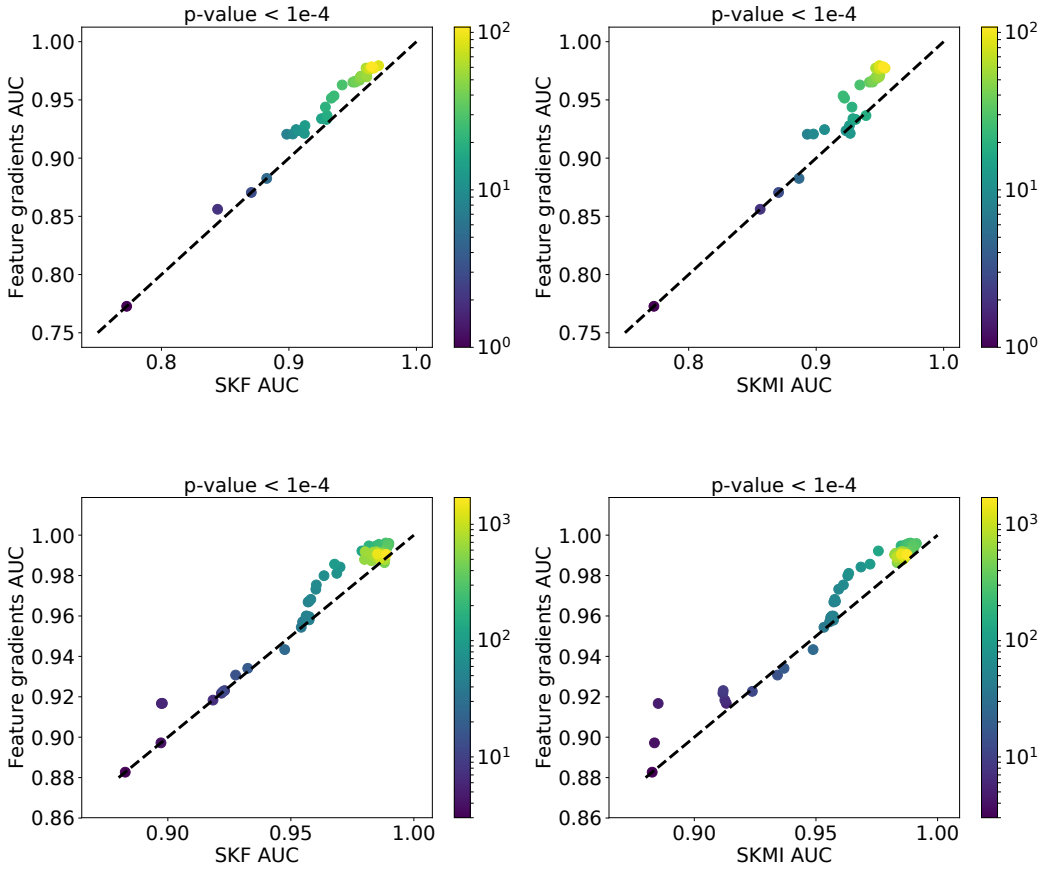

Figure 2: DFS order 6 performance vs. standard filter methods SKF (left) and SKMI (right) on *mnist35* (top) and *gisette* (bottom). Points above the dashed line indicate where DFS outperformed the competitor and vice versa below the dashed line. The color of the points denotes the feature subset size.

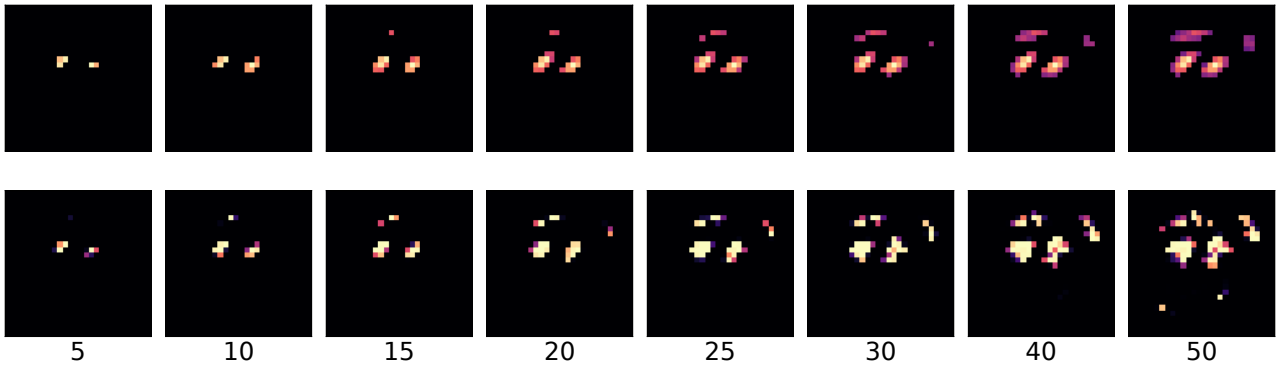

Figure 3: Feature selection on *mnist35*. The different columns represent different feature subset sizes (denoted below the bottom row). The features selected by the filter method utilizing the one-way ANOVA criterion (top) and by DFS order 6 (bottom) are colored according to their (normalized) scores.

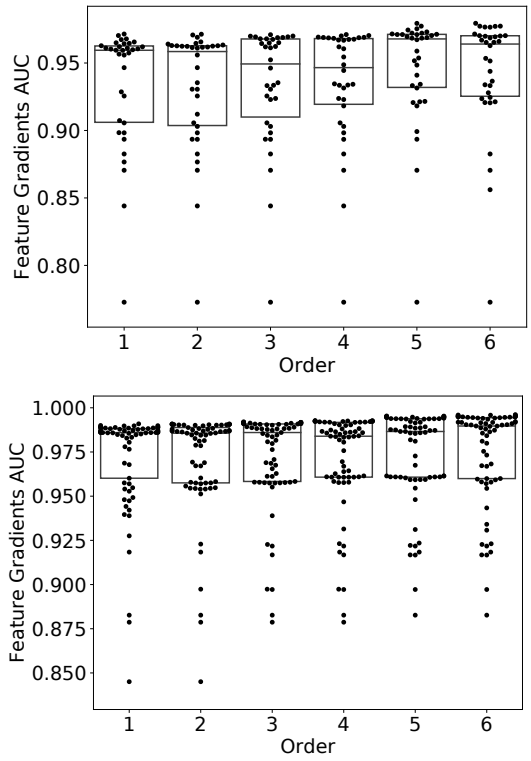

Figure 4: Scores (y-axis) as a function of DFS order (x-axis) on *mnist35* (top) and *gisette* (bottom). Each dot for a given order corresponds to a different number of features selected. The boxplots show median, upper and lower quartile of each distribution. The statistical significance of the improvement between orders 1 and 6 was measured with a paired t-test: for *mnist35*, the p-value is $< 10^{-4}$; for *gisette*, the p-value is 0.043.

In Figure 3, we highlight which features within the images are selected by the previous filter-based approach, SKF (top), and our algorithm (bottom). In particular, we plot values of the feature scores, *i.e.,* elements of the solution $s = \sigma(v)$ for different settings of the penalty λ corresponding to different amounts of sparsity. Across all the sparsity levels we considered, the selected features tended to cluster in the top half of the image. Indeed, it's easy to see that the most useful features to distinguish the digit "3" from the digit "5" are those located where the "upper loop" in the digit closes or opens. However, in contrast to the features selected by filter-based methods, which tend to concentrate owing to considering only single features at a time, the DFS-selected features can be more "diffuse" owing to the simultaneous consideration of multiple features which may not be "spatially close" but are more predictive of the target as a group.

Finally, we investigate whether increasing the order of the estimator produces significant improvements in accuracy. In Figure 4, we show the distribution of the AUCs for subsets of different sizes across different orders of the estimator. Overall, we see that higher orders correspond to better performance. We measure statistical significance with a paired t-test (paired across feature subsets sizes) and find that the improvement between order 6 and order 1 is statistically significant with a p-value of $< 10^{-4}$ for *mnist35* (top).

We repeat the same experiments for *gisette*, an even more challenging dataset in which the number of features is close to the number of samples available. The results in Figure 2 bottom left and right show that our method shows significantly better performance than the baselines we considered. As was the case for the previous dataset, increasing the order of DFS significantly increases performance (Figure 4 bottom). In general, we recommend using the highest computation-

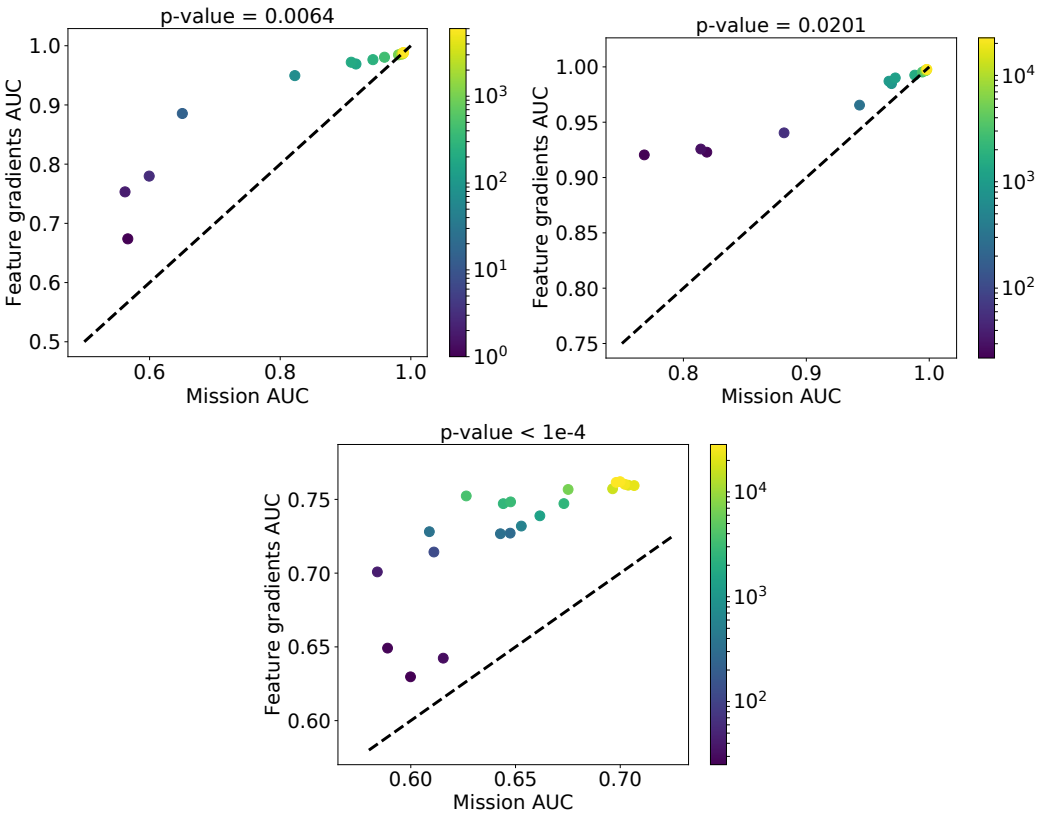

Figure 5: DFS order 4 performance vs. MISSION on *rcv1* (top left), *webspam* (top right), and *criteo* (bottom center).

ally feasible estimator order to maximize performance on a given task.

Large-scale experiments. Here, we compare DFS order 4 against MISSION on one moderate-sized dataset *rcv1* and two large datasets *webspam* and *criteo*. In the case of *rcv1*, a mini-batch size of 1000 was used. For *webspam*, the mini-batch size was 8, and for *criteo* it was 100. In these two latter cases, the mini-batch size was selected to be the largest that would fit into GPU memory on an NVIDIA Tesla P100. Also, in the latter two datasets, gradients were accumulated up to a size of 1000 examples. After training DFS for one epoch to learn the feature subsets, the MISSION code was used to train and evaluate an SGD classifier using the selected features. To support a fair comparison, the authors' code was modified to support this evaluation[4]. MISSION was then also run for the selected number of features.

Figure 5 shows the performance of DFS order 4 vs. MISSION on *rcv1* (top left) and *webspam* (top right). For both datasets, DFS is able to locate feature subsets resulting in test set performance significantly better than MISSION (significance level 0.01, with p-values of 0.0064 and 0.0201 for *rcv1* and *webspam*, respectively). Notably, for the webspam dataset, DFS order 4 is able to maintain 0.92 AUC on test with just 22 features selected out of over 16 million.

Finally, as shown in Figure 5 bottom, DFS significantly (p-value $< 10^{-4}$) outperforms MISSION on *criteo* across *all* the sparsity levels (*i.e.,* number of features) considered.

5 Conclusion

As the experiments show, our proposed method, Differentiable Feature Selection, results in feature subsets that are more predictive of the targets than standard filter-based methods on small datasets as well as state-of-the-art streaming feature selection on large datasets. The main limitation of DFS lies in the data centering step performed per mini-batch, which results in large memory utilization. In memory-constrained environments (*e.g.,* when using GPUs to accelerate computations), this issue can be mitigated by accumulating gradients. However, avoiding the data centering step

[4]Specifically, we modified the code to allow a set of features to be input prior to test.

entirely would result in better memory usage and faster computation, for example, when dealing with sparse data. We will focus on this aspect in future work.

Differentiable Feature Selection can be trivially extended to the multi-classification setting via the standard softmax relaxation applied across multiple outputs. Additionally, our method is also capable of "online" feature selection, where instances are presented one at a time. Preliminary results (not shown) indicate that DFS is capable of tracking which feature subsets are most correlated with the target as a function of time on *criteo*. A promising direction would be to combine DFS with dynamic feature engineering wherein features are iteratively constructed, rather than first expanded and then filtered, as done in this paper.

References

A. Abid, M. F. Balin, and J. Zou. Concrete autoencoders for differentiable feature selection and reconstruction. *arXiv preprint arXiv:1901.09346*, 2019.

A. Aghazadeh, R. Spring, D. LeJeune, G. Dasarathy, A. Shrivastava, and R. G. Baraniuk. MISSION: Ultra large-scale feature selection using count-sketches. *arXiv preprint arXiv:1806.04310*, 2018.

G. Forman. An extensive empirical study of feature selection metrics for text classification. *JMLR*, pages 1289–1305, 2003.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, pages 1157–1182, 2003.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, pages 273–324, 1997.

W. Kong and G. Valiant. Estimating learnability in the sublinear data regime. *CoRR*, abs/1805.01626, 2018.

H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. *CoRR*, abs/1806.09055, 2018.

C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, pages 227–234, 1995.

Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, pages 2507–2517, 2007.

Y. Sun, S. Todorovic, and S. Goodison. Local-learning-based feature selection for high-dimensional data analysis. *IEEE transactions on pattern analysis and machine intelligence*, pages 1610–1626, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, pages 267–288, 1996.

K. Yu, X. Wu, W. Ding, and J. Pei. Towards scalable and accurate online feature selection for big data. In *ICDM*, pages 660–669, 2014.

L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, pages 856–863, 2003.

L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *JMLR*, pages 1205–1224, 2004.