Scaling a Neyman-Pearson Subset Selection Approach Via Heuristics for Mining Massive Data

Gregory Ditzler*
Drexel University
Philadelphia, PA 19104 USA
gregory.ditzler@gmail.com

Matthew Austen
Rowan University
Glassboro, NJ 08028 USA
austen27@students.rowan.edu

Gail Rosen
Drexel University
Philadelphia, PA 19104 USA
gailr@ece.drexel.edu

Robi Polikar
Rowan University
Glassboro, NJ 08028 USA
polikar@rowan.edu

***Abstract*—Feature subset selection is an important step towards producing a classifier that relies only on relevant features, while keeping the computational complexity of the classifier low. Feature selection is also used in making inferences on the importance of attributes, even when classification is not the ultimate goal. For example, in bioinformatics and genomics feature subset selection is used to make inferences between the variables that best describe multiple populations. Unfortunately, many feature selection algorithms require the subset size to be specified a priori, but knowing how many variables to select is typically a nontrivial task. Other approaches rely on a specific variable subset selection framework to be used. In this work, we examine an approach to feature subset selection works with a generic variable selection algorithm, and our approach provides statistical inference on the number of features that are relevant, which may be unknown to the generic variable selection algorithm. This work extends our previous implementation of a Neyman-Pearson feature selection (NPFS) hypothesis test, which acts as a meta-subset selection algorithm. Specifically, we examine the conservativeness of the NPFS approach by biasing the hypothesis test, and examine other heuristics for NPFS. We include results from carefully designed synthetic datasets. Furthermore, we demonstrate the NPFS's ability to perform on data of a massive scale.**

***Keywords*-feature subset selection, Neyman-Pearson**

I. Introduction

Many datasets produced for data analysis, classification and regression are typically represented by a large number of variables. Such data observations could be a collection of variables extracted from sensors, robotics [1], or protein families [2]. Furthermore, it is well known that many of the variables (features) in a given dataset are uninformative for prediction on an outcome (i.e., the variable is irrelevant). Subset selection, a.k.a. feature or variable selection, is the process of determining a subset of the variables that are relevant for the task of prediction, regression, or knowledge discovery [3]. In this work, we are primarily interested in developing a meta-algorithm that is based on approaches that can score features independent of a classifier (i.e., filter methods), while providing statistical inference on the number of features that are important in a a dataset. Our approach allows users to freely choose a feature scoring objective function and a post-hoc hypothesis test is applied to detect feature importance. The resulting subset contains the variables that are *relevant*, while possibly penalizing the subset for containing variables that are *redundant*. The balance of relevancy and redundancy is controlled through the selection of a variable scoring function.

Feature subset selection algorithms typically fall into one of three broad categories: *wrapper-*, *embedded-*, and *filter-based* approaches. Wrapper-based subset selection algorithms optimize the feature set with respect to a specific classifier, and therefore is the most computationally complex. Embedded-based methods optimize the parameters of the classifier and variable selector simultaneously. Note that both of the two aforementioned methods are dependent on the optimization of a classifier. Filter-based variable selection methods score features by a function to determine importance that is independent of the error of a classifier. The advantage of a filter-based approach is that they are typically of much lower computational complexity than a wrapper or embedded methods.

In this work we assume that a dataset $\mathcal{D}$ contains observations $(\mathbf{x}_j, y_j)$, where $\mathbf{x}_j \in \mathcal{X}$ and $y_j \in \mathcal{Y}$. The vector $\mathbf{x}_j$ is a collection of random variables $\{X_1, \ldots, X_i, \ldots, X_K\}$, and many of the variables are assumed to carry little to no information on y_j; hence, marking the variable as irrelevant. Note that we have used the index j to index observations and i to index a variable (i.e., an index within $\mathbf{x}_j$). There exists an optimal number of relevant variables k^* that carry information on the outcome space $\mathcal{Y}$; however, k^* is unknown to any algorithm. The primary objective of this work is to determine k^* from a generic feature subset selection algorithm. We assume there is access to a variable selection algorithm, denoted by $\mathcal{A}$, that returns the k indices of the variables that are important, where $k < K$. Our objective is to examine sets of k indices and provide some inference – post feature subset selection – on the value of k^*.

Our previous work has addressed feature subset selection in such a setting; however, the conservativeness of the approach, which we call *N*eyman-*P*earson based *F*eature *S*election (NPFS), was not addressed. In this work, we examine biasing the post-hoc hypothesis test and perform a series of

G. Ditzler and G. Rosen are with the Dept. of Electrical & Computer Engineering at Drexel University, Philadelphia, PA. They are supported by the NSF CAREER award #0845827, and NSF Award #1120622.

M. Austen and R. Polikar are with the Dept. of Electrical & Computer Engineering at Rowan University. They are supported by the NSF Award #1310496.

*Corresponding author.

experiments on synthetic data to fully understand the dynamics of NPFS. Furthermore, we present new experiments and an open-source implementation of the scripts used to produced the results in this manuscript. We also demonstrate how NPFS can be applied to massive datasets, which is an important aspect of the flexibility of our approach.

This manuscript is organized as follows: Section II highlights some related works and the generic subset selection algorithms that can be used with the proposed approach. Section III presents our existing approach for subset selection using NPFS and the extensions we are proposing in this work (NPFS-β). Section IV compares NPFS-β to its baseline and existing subset selection algorithms. Section V draws conclusions and provides avenues for future research.

II. Related Work

The selection of the most informative variables is not only important to reduce the feature set to one that only contains relevant features [4], [3], but also one that can lead to a classifier/regressor with lower complexity (i.e., the VC-dimension of a function class $\mathcal{H}$ typically increases with the dimensionality) [5], [6], [7]. In addition to desirable theoretical properties, feature subset selection has been shown to be extremely helpful in many areas with traditionally large dimensional data, such as genomics [8], micro-array analysis [9], cancer classification [10], robotics [1], and many other fields.

Kohavi & John demonstrated that wrapper-based approaches can lead to classifiers that produce a small generalization loss [11], [12]; however, the Achilles' heal of these methods is the computational complexity, as the selected features are classifier dependent and a new classifier needs to trained for each selection. Recent efforts by Bolón-Canedo et al. have attempted to implement a distributed wrapper to improve the FS wrapper's run time [13], but their approach still needs to generate classifiers, which still leads to a computationally complex solution, and optimality of the wrapper cannot be guaranteed [14].

Embedded methods jointly optimize the classifier's parameters and feature selector simultaneously [4], [3]. The difference between embedded and wrapper based approaches, is that the FS for embedded methods is built into the objective function being optimized, whereas a wrapper is simply a search procedure for the best features given a classifier. For example, least absolute shrinkage and selection operator (LASSO) is a popular form of variable selection in regression setting, and used in times of linear prediction [15]. The central concept behind LASSO is the minimization of the ℓ^2-norm of the difference between the predictors $\mathbf{y}$ and the prediction $\hat{\mathbf{y}} = \mathbf{X}\theta$ with a penalization of the θ's ℓ^1-norm, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\mathsf{T}$. More formally, LASSO's objective is given by

$$\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\theta\|_2^2 + \lambda \|\theta\|_1 \tag{1}$$

where $\|\cdot\|_2$ and $\|\cdot\|_1$ represent the ℓ^2- and ℓ^1-norm, respectively, and $\lambda \geq 0$. The solution of equation (1) tends to be sparse because of the ℓ^1-norm minimization. Unfortunately, LASSO is not always well suited for problems where there are more data observations than variables. In such a setting, LASSO chooses all features. Elastic-net was developed to avoid this problem (see [16]) by changing the objective to

$$\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\theta\|_2^2 + \lambda_1 \|\theta\|_1 + \lambda_2 \|\theta\|_2^2 \tag{2}$$

However, even with subset selection methods such as elastic-net, the end user is limited to a specific loss function and choosing appropriate values for λ_1 and λ_2 is highly non-trivial[1]. It is also worth mentioning that LASSO and elastic-nets do not provide an avenue for any penalization of redundancy.

Recall that filter methods for feature subset selection decouple the objective of classification and feature selection. This class of FS methods score features on measures that are independent of the loss (performance) of a classifier (e.g., mutual information [18], [19], [20], [21], χ^2 statistics [22], correlation [23], or RELIEF's near-hit & near-miss [24]). As Brown et al. state, the *filter assumption* must be met for these approaches to be effective, i.e., the optimization of the feature set and classifier can be performed in two stages: (i) optimize the feature set, and (ii) using the result from (i) optimize the classifier. Greedy algorithms are often selected for optimizing the scoring function for filter methods. There exists a class of functions that are known as submodular [25], [26], which have some nice theoretical properties that can get ϵ-close to the optimal solution. Submodularity can be interpreted as a discrete analogue of convexity [25]. Submodular functions are becoming increasingly common in combinatorial optimization [27], [28], [29]; however, there is still the issue of when to stop adding to the subset size and one must show the function being used in subset selection is submodular.

III. Extending NPFS with Heuristics

In this section, we present the NPFS approach to feature selection with a post-hoc hypothesis that is applied after feature scoring and the proposed bias to the test.

A. Overview

We have recently proposed a post-hoc hypothesis test for detecting the number of important features in a dataset $\mathcal{D}$ using a post-hoc hypothesis test [30]. The central premise of NPFS is that a generic feature selection algorithm (e.g., mRMR) $\mathcal{A}$ selects k of K features on different bootstrap samples from $\mathcal{D}$; however, simply because $\mathcal{A}$ returned k features does not imply that all or relevant or that are more relevant features because k could be greater or less than k^*. A post-hoc hypothesis test is applied after calling $\mathcal{A}$ on the bootstrap datasets to detect the number of features that can be deemed important. NPFS an ideal candidate for practitioners of data mining tools to use

[1] The penalization of elastic-nets is sometimes written as $\alpha\|\theta\|_1 + (1-\alpha)\|\theta_1\|_2^2$ where $\alpha = \lambda_2/(\lambda_2 + \lambda_1) \in [0, 1]$ to simplify the problem. The problem of choosing an appropriate value for α is still not an easy task, and must be tuned (see [17] for an example of elastic-net and LASSO applied to metagenomic data).

NPFS with a generic subset selection tool (i.e., $\mathcal{A}$) that may not provide access to the objective function values.

Our previous work derived as relevant feature detector by forming a hypothesis test using Neyman-Pearson's lemma. Our method can be summarized in the following steps (see Figure 1 for a visual representation of NPFS).

1) Run $\mathcal{A}$ on n independently sampled datasets from $\mathcal{D}$. These sampled datasets can be a result of cross-validation or bootstrap samples[2]. Form a matrix $\mathbf{Z} \in \{0,1\}^{K\times n}$ where $\{\mathbf{Z}\}_{il}$ indicates if feature i was selected on trial l by $\mathcal{A}$.
2) Compute a threshold ζ_{crit} using (3), where z is a test statistic that is distributed according to a binomial distribution.
$$\mathbb{P}(z > \zeta_{\text{crit}}|H_0) = \alpha \tag{3}$$
where H_0 is the null hypothesis and α is the size of the hypothesis test.
3) Let $\{\mathbf{z}\}_i = \sum_{l=1}^{n} \{\mathbf{Z}\}_{il}$. If $\{\mathbf{z}\}_i > \zeta_{\text{crit}}$ then feature i belongs to the relevant set, otherwise the feature is marked not relevant. Use only the features selected by the Neyman-Pearson detector for learning a classification or regression function.

We refer to this approach as NPFS and we discuss the approach followed by our proposed modifications in the next section.

B. Biasing the Test Statistic

One of the issues that was observed with our method is that if n is increased NPFS may select more features than are actually relevant. We attribute this to have the null hypothesis of the NPFS set to assume that all features can fall into the relevant set uniformly at random which is an unrealistic assumption. One way to address this problem is to regularize the hypothesis with a bias term as follows:

$$H_0 : p_0 + \beta = p_1$$
$$H_1 : p_1 > p_0 + \beta$$

where H_0 is the null hypothesis, H_1 is the alternative hypothesis and $\beta \in (0, 1 - p_0]$ is a bias term. However, a practical selection of β should be small. This biased algorithm is referred to as NPFS-β. The effect of the β heuristic is examined in the experimental results section.

We need to show that NPFS-β has a nearly identical solution as NPFS. Beginning with the definition of the Neyman-Pearson likelihood test, we have:

$$\begin{aligned}\frac{\mathbb{P}(Z_n = z|H_1)}{\mathbb{P}(Z_n = z|H_0)} &= \frac{\binom{n}{z}p_1^z(1-p_1)^{n-z}}{\binom{n}{z}(p_0+\beta)^z(1-(p_0+\beta))^{n-z}} \\ &= \left(\frac{1-p_1}{1-(p_0+\beta)}\right)^n \cdot \left(\frac{p_1(1-(p_0+\beta))}{(p_0+\beta)(1-p_1)}\right)^z \\ &> \zeta_{\text{crit}}\end{aligned}$$

[2]Our software implementation uses the bootstrap.

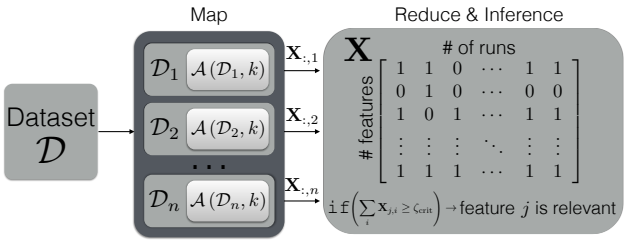


Fig. 1. Abstract diagram of the NPFS's implementation from a fundamental point-of-view. There are n bootstrap samples drawn from a distribution $\mathcal{D}$ and provided to a base selection algorithm $\mathcal{A}$. NPFS provides a post-hoc test to the predictions of results from the bootstrap experiments.

Observing that $\left(\frac{1-p_1}{1-(p_0+\beta)}\right)^n$ is simply a constant, which can be moved to the other side of the inequality, resulting in a new threshold ζ'_{crit}. Thus,

$$\left(\frac{p_1(1-(p_0+\beta))}{(p_0+\beta)(1-p_1)}\right)^z > \zeta'_{\text{crit}}$$

Taking the logarithm gives us

$$z \log\left\{\frac{p_1(1-(p_0+\beta))}{(p_0+\beta)(1-p_1)}\right\} > \zeta''_{\text{crit}}$$

where, again, the logarithm term is simply a constant and it can be removed to find a scaled threshold ζ'''_{crit}. Thus, we are seeking

$$z > \zeta'''_{\text{crit}}$$

where ζ'''_{crit} is a critical threshold determined by $\mathbb{P}(z > \zeta'''_{\text{crit}}|H_0) = \alpha$ (recall that by definition z is the sufficient statistic). Since the probability distribution on the null hypothesis is known (i.e., Binomial with probability $p_0 + \beta$), we may explicitly solve for ζ_{crit}.

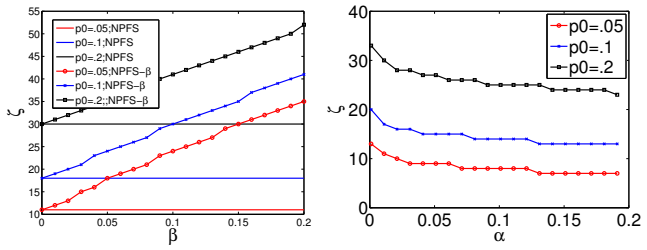
$$\mathbb{P}(z > \zeta'''_{\text{crit}}|H_0) = 1 - \mathbb{P}(z \leq \zeta'''_{\text{crit}}|H_0) = \alpha$$

Since $\mathbb{P}(z \leq \zeta'''_{\text{crit}}|H_0)$ has a closed form expression it can be obtained from a lookup table. The solution to NPFS-β is nearly identical to NPFS. The only difference is the probability under the null hypothesis.

C. The Effect of NPFS-β

Figure 2(a) shows the effect that β has on the critical threshold ζ'''_{crit} for a fixed value of n and size of the hypothesis test (for convienence we write ζ'''_{crit} as ζ). For this experiment $n = 100$ and $\alpha = 0.01$. As desired increasing β increases the critical threshold; hence, making it more difficult to select a feature as being relevant, which leads to a more conservative test for selecting features. Blindly increasing β to be arbitrarily large (e.g., $\beta = 0.3$) more than doubles the number of times a feature needs to be detected as relevant by $\mathcal{A}$ to be considered relevant by NPFS-β.

However, why not tune the value of α, which can also be used to control the size/conservativeness of the test? Figure 2(b) show α varying from 0.001 to 0.2. This figure shows that we have more control over how conservative the test is by



(a) ζ for NPFS vs. NSFS-β (b) Effect of α

Fig. 2. (**left**) Increasing the value of β increases the critical threshold ζ for several different values of p_0 ($n = 100$ and $\alpha = 0.01$ for this experiment). (**right**) Effect that α has on the critical threshold ζ.

changing β than α. Furthermore, β, like α, has an intuitive sense of control over the conservativeness of the test. Also, α, as we would expect, has the most control of ζ when α is small. Unfortunately, this makes it more difficult to control the conservatives of the test when such small changes in α correspond to exponential changes in ζ.

D. Early Stopping

NPFS requires that the number of bootstraps, or random calls to a collection of databases, be known in advance. Therefore, in this section we would like to examine how to perform an implementation of early stopping of NPFS when it can be stated that the change in the importances probabilities is miniscule. Let $\hat{\mathbf{p}}_t \in [0,1]^K$ be a vector containing the ratio of the number of times a feature was detected as relevant over the number of trials that have been performed until time t. A straightforward stopping criteria is to define a bound on the average difference between two consecutive time stamps t and $t-1$, which is given by

$$\frac{1}{K}\|\hat{\mathbf{p}}_t - \hat{\mathbf{p}}_{t-1}\|_1 \leq \xi \tag{4}$$

where $\xi > 0$ is some small number to define the minimal change needed to continue running NPFS. While the selection ξ must be chosen somewhat carefully, we argue that this stopping criteria is a quick way to perform early stopping and has been used for early stopping in neural network training [31].

IV. Experiments

We evaluated NPFS with the bias parameter on carefully designed synthetic datasets so that we can understand how we should expect NPFS to react under different conditions (e.g., number of features, number of relevant features, etc.). A Matlab implementation of NPFS is released under the GNU GPL v3.0, which can be found at https://github.com/EESI/NPFS.

We begin our evaluation of NPFS on synthetic data so that we may design experiments to determine how we expect the algorithm to react in different settings. The synthetic data is generated as follows: M observations with features that are independent and identically distributed (iid) uniform random variables in the interval $[0, 10]$. Each feature vector $\mathbf{x}_j$ for $j \in [M]$ has K features. The true labeling function, unknown to any algorithm, is given by,

$$y_j = \begin{cases} 1, & \sum_{i=1}^{k^*} \mathbf{x}_j(i) \leq 5 \cdot k^* \\ 0, & \text{otherwise} \end{cases}$$

Hence, only the first k^* features carry information for determining the label y_j of a feature vector $\mathbf{x}_j$. Our goal is to identify, using the proposed NPFS approach, those features (indices $i \in [k^*]$) that are relevant to the classification problem. We can also use this process to generate a massive amount of data to evaluate our algorithm on large scale datasets (e.g., our experiments include a 58M+ observation dataset).

A. Variation in β and n

We begin by examining the effects of β and n on NPFS when evaluated on the synthetic dataset described above. In this experiment $\beta \in \left[0, \frac{1}{5}\right]$, $n \in [10, 500]$, $M = 1,000$, $K = 50$, and mutual information maximization (MIM) is the base feature subset selection method $\mathcal{A}$ [22]. The Jaccard index is our performance figure of merit, which is given by:

$$\mathrm{J}(A, B) = \frac{|A \cap B|}{|A \cup B|} \in [0, 1]$$

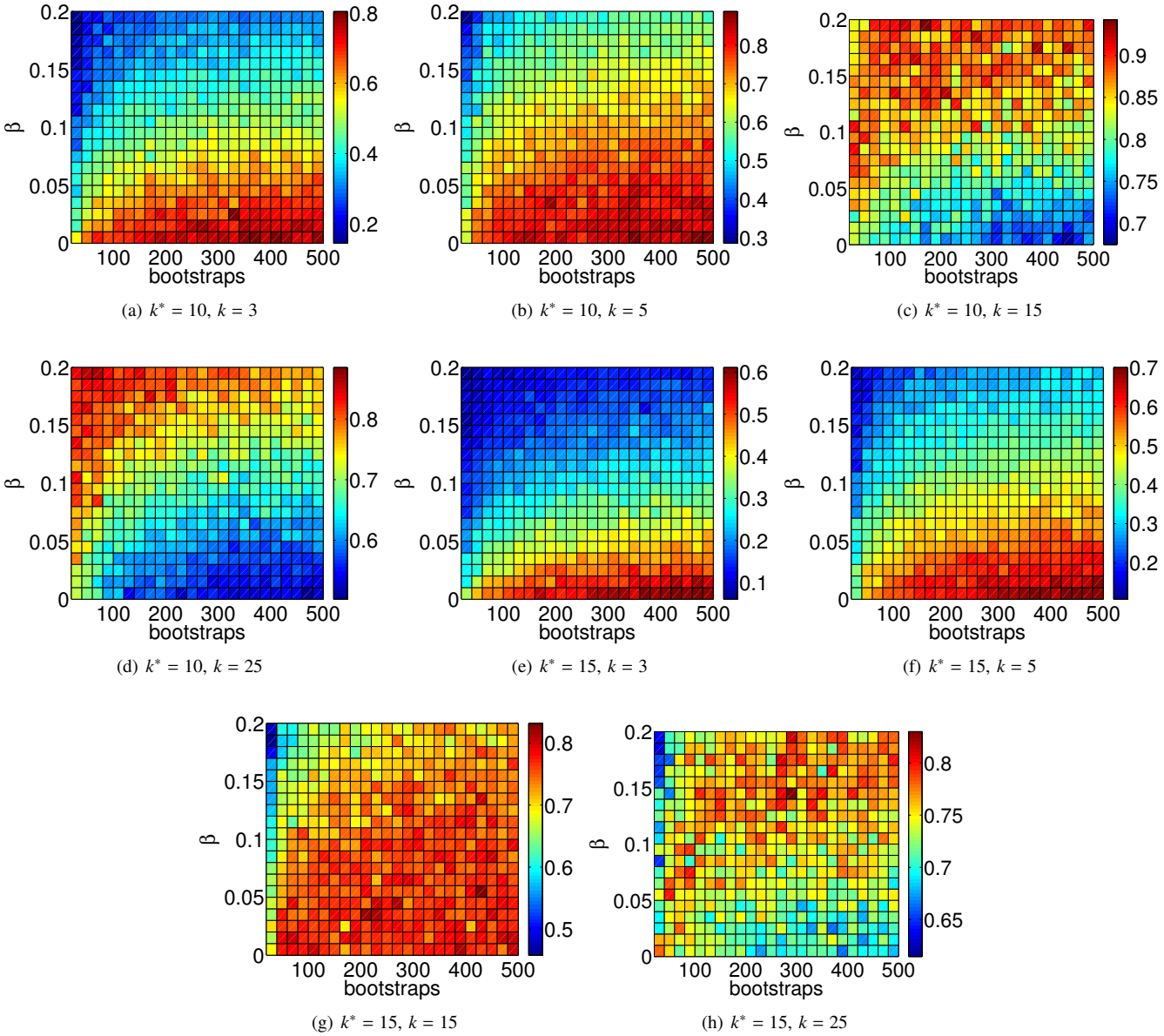
where A is the set of features detected by NPFS, and B is the set $\{1, \ldots, k^*\}$. Figure 3 shows the Jaccard index of NPFS evaluated with different values of k and k^*.

Figure 3(a), 3(b), 3(c), and 3(d) fixes the relevant feature set to $\{1 \ldots, 10\}$ and MIM selects 3, 5, 15, 25 features, respectively. Note that in the case of $k = \{3, 5\}$, MIM selects fewer than the optimal number of features (i.e., $k^* = 10$), and for $k = \{15, 25\}$, MIM selects more than the optimal number of features. NPFS works better with $\beta = 0$ when $k < k^*$, which goes along with our intuition about how β affects NPFS. β increases the critical threshold for the hypothesis test, which makes it *more difficult* for a feature to be detected as relevant; hence, the lower Jaccard indices for larger values of β in the setting where $k < k^*$. When $k > k^*$, we observe that β improves the Jaccard index for NPFS. Similar observations can be made for the remaining subplots in Figure 3, which shows the same experiment for $k^* = 15$.

Clearly, β can have a very large impact on NPFS even on this synthetic dataset, how should we proceed to choose β? In most settings, empirical results suggest to use a smaller value of k with a small $\beta < \frac{k}{K}$ then increase the number of bootstraps. This observation can be made in Figures 3(a), 3(b), 3(e), 3(f), and 3(g). These experiments demonstrate that β must be chosen very carefully and not naïvely, similar to choosing the learning rate for gradient descent (see Duda et al.'s discussion on learning rates and overshooting [32]).

B. Scalability for Massive M + Large K

We have also evaluated NPFS on a massive dataset to prove its scalability for datasets that have a massive M and a large K (i.e., $M = 58,340,000$ and $K = 10,000$). The entire dataset was separated into many smaller files (≈100MB) to allow for our computing nodes to load portions of the data into memory

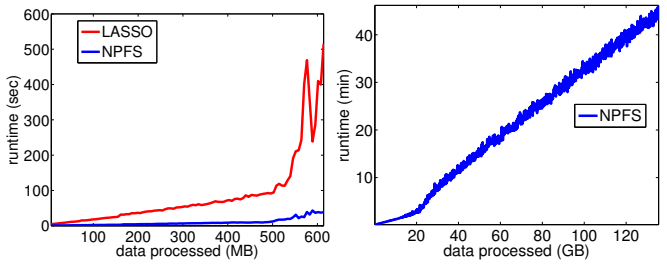


(a) $k^* = 10$, $k = 3$ (b) $k^* = 10$, $k = 5$ (c) $k^* = 10$, $k = 15$

(d) $k^* = 10$, $k = 25$ (e) $k^* = 15$, $k = 3$ (f) $k^* = 15$, $k = 5$

(g) $k^* = 15$, $k = 15$ (h) $k^* = 15$, $k = 25$

Fig. 3. Jaccard index of NPFS measured against the set $\{1, \ldots, k^*\}$ on a synthetic uniform dataset. The number of bootstraps are varied from $10 \rightarrow 100$ and the bias parameter is varied from $0 \rightarrow \frac{1}{5}$.



(a) NPFS vs. LASSO (b) NPFS on BigData

Fig. 4. (**left**) Runtime of NPFS versus LASSO on a large synthetic dataset. (**right**) NPFS evaluated on a very large dataset. The NPFS times can be interpreted as the amount of time it takes to complete a dataset of size XGB.

without crashing[3]. LASSO and NPFS were implemented in Matlab, where NPFS uses MIM as the base-subset selection algorithm. NPFS takes advantage of parallel computing in its implementation. We process about 100MB of data at a time. We also increase the number of bootstraps as the larger datasets are processed to take advantage of parallelism within NPFS.

Figure 4(a) shows the evaluation time of LASSO and NPFS on the synthetic dataset with increasing size. Clearly, NPFS is a far better performer in terms of the evaluation time as the size of the dataset increases, and the size of the datasets being

[3]The entire dataset is nearly 100GB, which is far larger than the memory on any of our computing nodes.

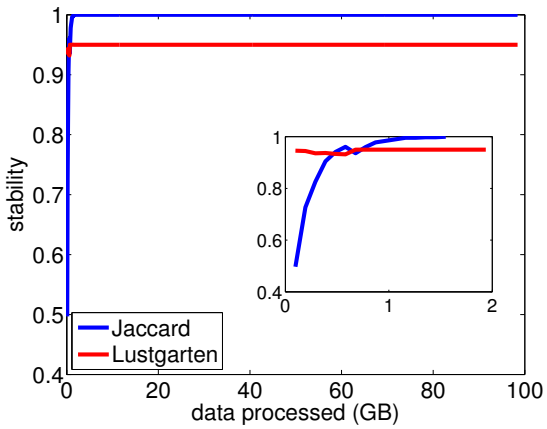


Fig. 5. Stability of NPFS on a massive amount of data using the Jaccard and Lustgarten indices as figures of merit. The inner plot shows the Jaccard and Lustgarten indices when processed over 2GB of data.

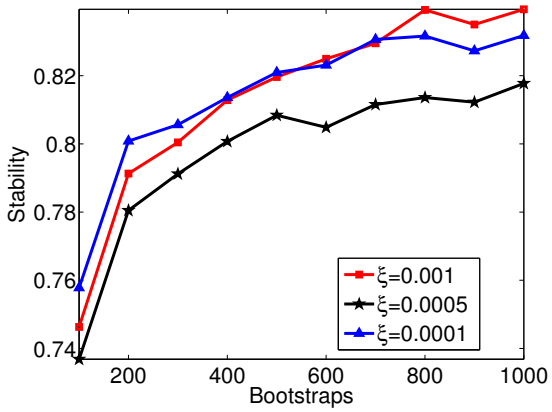


Fig. 6. The Jaccard index for $\xi = \{0.001, 0.0005, 0.0001\}$.

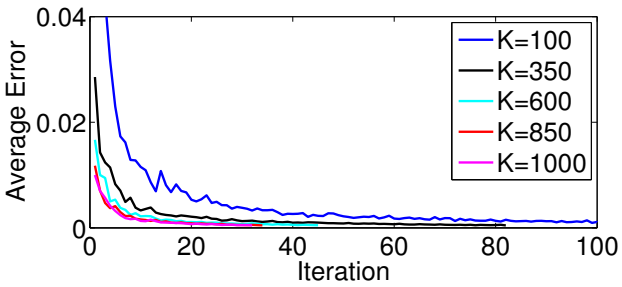


Fig. 7. The early stopping criteria, $\frac{1}{K}\|\hat{\mathbf{p}}_t - \hat{\mathbf{p}}_{t-1}\|_1$, for different size feature sets.

tested are – at the moment – not too large. This figure shows that is difficult to make LASSO perform well in a reasonable amount of time with a large dataset (i.e., larger than 1GB). Figure 4(b) shows NPFS applied on a 100GB dataset; however, we did not include LASSO because the software implementation would not run on datasets over ≈1GB. We observe that NPFS's runtime is increasing approximately linearly with the size of the data it is processing. In the previous example, LASSO saw a large jump in the runtime when the dataset became quite large in size, which could be attributed to LASSO being a polynomial time algorithm. Note that NPFS times could be further improved upon by increasing the number of parallel workers; however, due to software limitations, we must limit the number of parallel processes NPFS can use at once.

We also examined the stability of NPFS in the previous experiment using the Jaccard and Lustgarten indices, the latter of which is given by:

$$\mathsf{L}(A,B) = \frac{|A \cap B| - \frac{|A|\cdot|B|}{K}}{\min(|A|,|B|) - \max(0, |A| + |B| - K)}$$

The Lustgarten is a variation of Kuncheva's consistency index [33]; however, Lustgarten's index does not require that $|A| = |B| = k$, which is assumed by Kuncheva. Figure 5 shows the Jaccard and Lustgarten index of NPFS as data are processed. It is obvious that NPFS can detect the optimal feature set after a very small portion of the dataset was processed.

C. Early Stopping

We evaluated implementation of early stopping on the synthetic dataset described in the beginning of section IV, where $K = 100$, $k^* = 25$, $k = 10$, and we average our results over 100 randomly sampled datasets. We set the maximum number of bootstraps to $n = 1000$, which given the properties of the Binomial RVs in $\mathbf{Z}$, such a value of n should be sufficient to reach convergence as shown by Ditzler et al. [30]. We set $\xi = \{0.001, 0.0005, 0.0001\}$ and evaluated NPFS using the Jaccard index.

Figure 6 shows the Jaccard index for NPFS as a function of the number of bootstraps. We find that if early stopping is applied that the average stopping times for the three different thresholds are given by $T_{\xi=0.001} = 81.56$, $T_{\xi=0.0005} = 156.06$, and $T_{\xi=0.0001} = 758.12$. Similar to other areas that use early stopping, there are generally small decreases in performance; however, the advantage is the amount of time the algorithm takes to run is significantly lower with early stopping. NPFS was observed to converge in the Jaccard and Lustgarten rather quickly, and this can be explained by the quick convergence of $\frac{1}{K}\|\hat{\mathbf{p}}_t - \hat{\mathbf{p}}_{t-1}\|_1$ to zero. Figure 7 shows $\frac{1}{K}\|\hat{\mathbf{p}}_t - \hat{\mathbf{p}}_{t-1}\|_1$ for $\xi = 0.0005$ applied to varying levels of dimensionality. For all selections of K, NPFS stops running before the maximum number of bootstraps is reached when early stopping is implemented.

V. Conclusion

In this work we examined several extensions to the recently presented Neyman-Pearson Feature Selection (NPFS) approach that works with a generic subset selection algorithm to detect the optimal number of relevant feature set size. One of the key advantages of NPFS is that it is highly parallelizable and can, therefore, be applied to massive datasets in a relatively small amount of time. We examined biasing the Neyman-Pearson hypothesis test to reduce false positives and a method of early stopping. The primary observations that we found are:

- Early stopping can be quite effective at reducing the number of rounds that NPFS requires; however, the selection of ξ can degrade the performance of stability indices. Furthermore, we lose the parallelism by needing to check for convergence of $\|\hat{\mathbf{p}}_t - \hat{\mathbf{p}}_{t-1}\|_1$.
- Biasing the hypothesis test in NPFS by some amount β can reduce the false positives when k was chosen larger than k^*. The best suggestion for choosing k and β would be to set k small with a small value of β. Increasing the bootstraps then tended to detect k^*.
- NPFS can easily be scaled to massive datasets, which may not have been possible to process with traditional feature subset selection algorithms.

Our future work includes feature subset selection algorithms that can process massive datasets without considering all K features on each evaluation. One promising avenue of research is bandit-based subset selection algorithms, which was presented by Gaudel & Sebag [34].

References

[1] T. Nguyen, Z. Li, T. Silander, and T.-Y. Leong, "Online feature selection for model-based reinforcement learning," in *International Conference on Machine Learning*, 2013.

[2] Y. Lan, A. Kriete, and G. Rosen, "Selecting age-related functional characteristics in the human gut microbiome," *Microbiome*, vol. 1, no. 2, 2013.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[4] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications*. Springer, 2006.

[5] M. Kearns, *Computational Complexity of Machine Learning*. MIT Press, 1990.

[6] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[7] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verilag, 2nd ed., 1999.

[8] Y. Saeys, I. Inza, and P. Larra naga, "A review of feature selection techniques in bioinformatics," *Oxford Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[9] C. Lazar, J. Taminau, S. Meganck, D. Steenhoff, A. Coletta, C. Molter, V. de Schaetzen, R. Duque, H. Bersini, and A. Nowé, "A survey on filter techniques for feature selection in gene expression microarray analysis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1106–1119, 2012.

[10] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, pp. 389–422, 2002.

[11] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, pp. 273–324, 1997.

[12] G. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *International Conference on Machine Learning*, 1994.

[13] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A distributed wrapper approach for feature selection," in *European Symposiun on Artificial Neural Networks*, pp. 173–178, 2013.

[14] I. Tsamardinos and C. Aliferis, "Towards principled feature selection: relevancy, filters and wrappers," in *International Workshop on Artificial Intelligence and Statistics*, 2003.

[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of Royal Statistics Society*, vol. 58, no. 1, pp. 267–288, 1996.

[16] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society*, vol. 67, no. 2, pp. 301–320, 2005.

[17] D. Knights, E. K. Costello, and R. Knight, "Supervised classification of human microbiota," *FEMS Microbiology Reviews*, vol. 35, no. 2, pp. 343–359, 2011.

[18] W. Duch, K. Grąbczewski, T. Winiarski, J. Biesiada, and A. Kachel, "Feature selection based on information theory, consistancy, and separability indices," in *International Conference on Neural Information Processing*, pp. 1951–1955, 2002.

[19] H. Almuallim and T. Dietterich, "Efficient algorithms for identifying relevant features," in *Canadian Conference on Artificial Intelligence*, 1992.

[20] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *Journal of Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[21] G. Brown, "A new perspective for information theoretic feature selection," in *International Conference on Artificial Intelligence and Statistics*, pp. 49–56, 2009.

[22] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of the Workshop on Speech and Natural Language*, pp. 212–217, 1992.

[23] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *International Conference on Machine Learning*, 2003.

[24] K. Kira and L. Rendell, "A practical approach to feature selection," in *National Conference on Artificial Intelligence*, 1992.

[25] S. Iwata, L. Fleischer, and S. Fujishige, "A combinatorial strongly polynomial algorithm for minimizing submodular functions," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pp. 96–107, 2000.

[26] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000.

[27] A. Das and D. Kempe, "Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection," in *International Conference on Machine Learning*, 2011.

[28] Y. Liu, K. Wei, K. Kirchoff, Y. Song, and J. Bilmes, "Submodular feature selection for high-dimensional acoustic score spaces," in *International Conference on Acoustics, Speech and Signal Processing*, pp. 7184–7188, 2013.

[29] A. Krause and V. Cevher, "Submodular dictionary selection for sparse representation," in *International Conference on Machine Learning*, 2010.

[30] G. Ditzler, R. Polikar, and G. Rosen, "A bootstrap based neyman–pearson test for identifying variable importance," *IEEE Transactions on Neural Networks and Learning Systems*, 2014.

[31] S. Haykin, *Neural Networks and Learning Machines*. Pearson, 2009.

[32] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, Inc., 2nd ed., 2001.

[33] L. I. Kuncheva, "A stability index for feature selection," in *International Conference on Artifical Intelligence and Application*, pp. 390–395, 2007.

[34] R. Gaudel and M. Sebag, "Feature selection as a one-player game," in *International Conference on Machine Learning*, 2010.