A Feature Selection Algorithm Capable of Handling Extremely Large Data Dimensionality

Yijun Sun[*] Sinisa Todorovic[†] Steve Goodison[‡]

Abstract

With the advent of high throughput technologies, feature selection has become increasingly important in a wide range of scientific disciplines. We propose a new feature selection algorithm that performs extremely well in the presence of a huge number of irrelevant features. The key idea is to decompose an arbitrarily complex nonlinear models into a set of locally linear ones through local learning, and then estimate feature relevance globally within a large margin framework. The algorithm is capable of processing many thousands of features within a few minutes on a personal computer, yet maintains a close-to-optimum accuracy that is nearly insensitive to a growing number of irrelevant features. Experiments on eight synthetic and real-world datasets are presented that demonstrate the effectiveness of the algorithm.

1 Introduction

With the advent of high throughput technologies, feature selection has become increasingly important in a wide range of scientific disciplines. Its goal is to extract the most relevant information about each observed datum from a potentially overwhelming quantity of features to facilitate the underlying data analysis. In this paper, we consider feature selection for the purposes of data classification. Not only can its proper design reduce system complexity and processing time, but it can also enhance system performance in many cases.

Existing feature selection algorithms are traditionally categorized as wrapper or filter methods [1]. In wrapper methods, a classification algorithm is employed to evaluate the goodness of a selected feature subset, whereas in filter methods criterion functions evaluate feature subsets by their information content, instead of optimizing the performance of any specific classification algorithm directly. Hence, filter methods are computationally much more efficient, but usually perform worse than wrapper methods. One major issue with wrapper methods is their high computational complexity. Many heuristic algorithms (e.g., forward and backward selection [2]) have been proposed to alleviate the computational issue. In the presence of tens of thousands features, a hybrid approach is usually adopted, wherein the number of features is first reduced by using a filter method, and then a wrapper method is used on the reduced feature set. Nevertheless, it still may take several hours to perform the search, depending on the classifier used in the wrapper method. Another issue with a wrapper method is its capability to perform feature selection for multiclass problems. To a large extent, this property depends on the capability of a classifier used in a wrapper method to handle multiclass problems. In many cases, a multiclass problem is first decomposed into several binary ones by using an error-correct-code method [3], and then feature selection is performed for each binary problem. This strategy further increases the computational burden of a wrapper method. One issue that is rarely addressed in the literature is algorithm implementation. Wrapper methods require the train of a large number of classifiers and manually specification of many parameters. This makes their implementation and use rather complicated, demanding an expertise in machine learning.

Embedded methods have recently received an increased interest (see, for example, [4] and the references therein.). In contrast to wrapper methods, embedded methods incorporate feature selection directly into the learning process of a classifier. A feature weighting strategy is usually adopted that uses real-valued numbers, instead of binary ones, to indicate the relevance of features in a learning process. This strategy has many advantages. For example, there is no need to pre-specify the number of relevant features. Also, standard optimization techniques (e.g., gradient descent) can be used to avoid combinatorial search. Hence, embedded methods are usually computationally more tractable than wrapper methods. Still, computational complexity is a major issue when the number of features becomes excessively large. Other issues, such as algorithm implementation and extension to multiclass problems, remain.

In this paper, we propose a new feature selection algorithm that addresses many aforementioned issues with prior work, including the problems with computational complexity, solution accuracy, algorithm implementation, capability to handle an extremely large number of features, and exten-

[*] Interdisciplinary Center for Biotechnology Research, University of Florida, Gainesville, FL 32611. Email: sunyijun@biotech.ufl.edu

[†] Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801. Email: sintod@uiuc.edu

[‡] Department of Surgery, University of Florida, Jacksonville, FL 32209. Email: steve.goodison@jax.ufl.edu

sion to multiclass problems. The key idea is to decompose an arbitrary complex, nonlinear model into a set of locally linear ones through local learning, and then estimate the relevance of features globally within a large margin framework. We demonstrate that through linearization the feature selection problem can be easily solved by using well-established machine learning and numerical analysis techniques.

1.1 Related Work Our approach is motivated by the ideas implemented in the RELIEF algorithm [5, 6]. RELIEF has been long regarded as a heuristic filter method, until recently we mathematically proved that RELIEF is an online algorithm that solves a convex optimization problem aimed at maximizing the average margin [7]. One major problem with RELIEF is that the nearest neighbors of a given sample are predefined in the original feature space, which typically yields erroneous nearest hits and misses in the presence of copious irrelevant features. The idea of using local information for estimating feature relevance is also exploited in the Simba algorithm [8]. One major problem with Simba is its implementation. The objective function optimized by Simba is characterized by many local minima. Also, Simba represents a constrained nonlinear optimization problem that cannot be easily solved by conventional optimization techniques. We empirically find that Simba performs well when the number of irrelevant features is small, but fails completely when there exist a relatively large number of irrelevant features (say 500). One possible explanation is that the chance for Simba to be stuck into local minima is increased dramatically with the number of features. Based on our mathematical analysis of RELIEF, we have recently proposed a new feature weighting algorithm referred to as I-RELIEF that performs significantly better than the above-mentioned algorithms [7]. However, as with all other algorithms in the RELIEF family, the objective function optimized by I-RELIEF is not directly related to the classification performance of a learning algorithm. Moreover, unlike the proposed algorithm, both RELIEF and I-RELIEF cannot provide a sparse solution.

The idea of using a large margin algorithm for feature selection is not new. For example, Weston et. al propose to perform feature selection directly in the SVM formulation, where the scaling factors are adjusted using the gradient of a (loose) theoretical upper bound on the error rate [9]. SVM-RFE is a well-known algorithm specifically designed for large-scale feature selection problems [10]. It works by iteratively training a SVM classifier with the current set of features and heuristically removing the features with small feature weights. As with wrapper methods, the structural parameters of SVM (i.e., the regularization and kernel parameters) may need to be re-estimated using cross-validation during the iterations. Also, a linear kernel is usually used. From our personal communication with the authors, it may be computationally expensive to use SVM-REF with a nonlinear kernel for high-dimensional data. ℓ_1-SVM with a linear kernel [11], with a proper parameter tuning, can lead to a sparse solution, where only relevant features receive nonzero weights. A similar algorithm is logistical regression with ℓ_1 regularization. It has been proved that ℓ_1 regularized logistical regression has a logarithmical sample complexity with respect to the number of irrelevant features [12]. The logarithmic dependence on the input dimension matches the best known bounds proved in various feature selection contexts [12, 13]. However, the linearity assumption of data models limits their application to general problems.

2 Our Algorithm

In the section, we present the detailed formulation of our feature selection algorithm. We also analyze the convergence and computational complexity of our algorithm, as well as present its extension to multiclass problems.

Let $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^J \times \{\pm 1\}$ be a training dataset, where $\mathbf{x}_n$ is the n-th data sample containing J features, and y_n is its corresponding class label. For clarity, we here consider only binary problems, while in Sec. 2.3 we generalize our algorithm to address multiclass problems. We first define the margin. Given a distance function, we find two nearest neighbors of each sample $\mathbf{x}_n$, one from the same class (called *nearest hit* or NH), and the other from the different class (called *nearest miss* or NM) [5]. The margin of $\mathbf{x}_n$ is then defined as $\rho_n = d(\mathbf{x}_n, \mathrm{NM}(\mathbf{x}_n)) - d(\mathbf{x}_n, \mathrm{NH}(\mathbf{x}_n))$, where $d(\cdot)$ is the distance function. For the purpose of this paper, we use the block distance to define a sample's margin and nearest neighbors, while other standard definitions may also be used. An intuitive interpretation of this margin is a measure as to how much $\mathbf{x}_n$ can "move" in the feature space before being misclassified. By the large margin theory [14], a classifier that minimizes a margin-based error function usually generalizes well on unseen test data. One natural idea then is to scale each feature, and thus obtain a weighted feature space, parameterized by a nonnegative vector $\mathbf{w}$, so that a margin-based error function in the *induced* feature space is minimized. The margin of $\mathbf{x}_n$, computed with respect to $\mathbf{w}$, is given by:

$$\rho_n(\mathbf{w}) = d(\mathbf{x}_n, \mathrm{NM}(\mathbf{x}_n)|\mathbf{w}) - d(\mathbf{x}_n, \mathrm{NH}(\mathbf{x}_n)|\mathbf{w}) \ . \tag{2.1}$$

By defining $\mathbf{z}_n = |\mathbf{x}_n - \mathrm{NM}(\mathbf{x}_n)| - |\mathbf{x}_n - \mathrm{NH}(\mathbf{x}_n)|$, where $|\cdot|$ is an element-wise absolute operator, $\rho_n(\mathbf{w})$ can be simplified as:

$$\rho_n(\mathbf{w}) = \mathbf{w}^T \mathbf{z}_n \ , \tag{2.2}$$

which is a linear function of $\mathbf{w}$ and has the same form as the sample margin defined in SVM using a kernel function. An important difference, however, is that by construction the magnitude of each element of $\mathbf{w}$ in the above margin definition reflects the relevance of the corresponding feature in a learning process. This is not the case in SVM except when a linear kernel is used, which however can capture only linear discriminant information. Note that the margin thus defined requires only information about the neighborhood of $\mathbf{x}_n$, while no assumption is made about the underlying data distribution. This means that by local learning we can transform an *arbitrary* complex, nonlinear problem into a set of locally linear problems.

The local linearization of a nonlinear problem allows us to avoid the computational difficulties of prior work. It also facilitates the mathematical analysis of our algorithm. The main problem with the above margin definition, however, is that the nearest neighbors of a given sample are unknown before learning. In the presence of thousands of irrelevant features, the nearest neighbors defined in the original space can be completely different from those in the induced space. To account for the uncertainty in defining local information, we develop a probabilistic model where the nearest neighbors of a given sample are treated as latent variables. Following the principles of the expectation-maximization algorithm [15], we estimate the margin through taking the expectation of $\rho_n(\mathbf{w})$ by averaging out the latent variables:

(2.3)

$$\begin{aligned}\bar{\rho}_n(\mathbf{w}) &= \mathbf{w}^T \Big(\textstyle\sum_{i\in\mathcal{M}_n} P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w})|\mathbf{x}_n-\mathbf{x}_i| \\ &\qquad - \textstyle\sum_{i\in\mathcal{H}_n} P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})|\mathbf{x}_n-\mathbf{x}_i|\Big) \\ &= \mathbf{w}^T\bar{\mathbf{z}}_n \ ,\end{aligned}$$

where $\mathcal{M}_n = \{i : 1 \le i \le N, y_i \neq y_n\}$, $\mathcal{H}_n = \{i : 1 \le i \le N, y_i = y_n, i \neq n\}$, $P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w})$ and $P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})$ are the probabilities that sample $\mathbf{x}_i$ is the nearest miss or hit of $\mathbf{x}_n$, respectively. These probabilities are estimated through the standard kernel density estimation method:

(2.4)

$$P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w}) = \frac{k(\|\mathbf{x}_n-\mathbf{x}_i\|_{\mathbf{w}})}{\sum_{j\in\mathcal{M}_n} k(\|\mathbf{x}_n-\mathbf{x}_j\|_{\mathbf{w}})} \ , \forall i{\in}\mathcal{M}_n \ ,$$

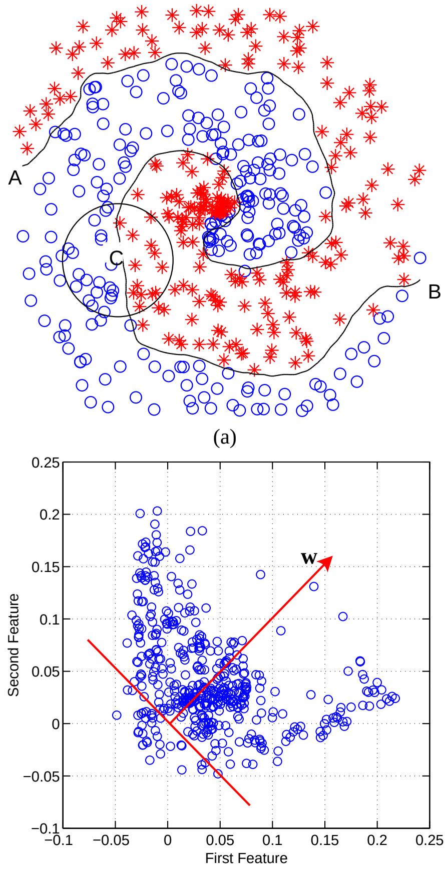
and

(2.5)

$$P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w}) = \frac{k(\|\mathbf{x}_n-\mathbf{x}_i\|_{\mathbf{w}})}{\sum_{j\in\mathcal{H}_n} k(\|\mathbf{x}_n-\mathbf{x}_j\|_{\mathbf{w}})} \ , \forall i{\in}\mathcal{H}_n \ ,$$

where $k(\cdot)$ is a kernel function. Specifically, we use the exponential kernel $k(d) = \exp(-d/\sigma)$, where the kernel width determines the resolution at which the data is locally analyzed.

To motivate the above formulation, we consider the well-known Fermat's problem where the two-class samples are distributed in a two-dimensional space, forming a spiral shape, as illustrated in Fig. 1a. A possible decision boundary is also plotted. If one walks from point A to B along the decision boundary, at any given point (say, point C), one obtains a liner problem locally. One possible linear formulation is given in Eq. (2.3). It is clear that for the spiral problem, both features are equally important. By projecting the transformed data $\bar{\mathbf{z}}_n$ onto the feature weight vector $\mathbf{w} = [1, 1]^T$, we observe that most samples have positive margins (Fig. 1b). The above arguments generally hold for arbitrary nonlinear problems for a wide range of values of kernel width, as long as the local linearity condition is preserved. We will shortly see that the performance of our algorithm is indeed robust against this parameter.



(a)

(b)

Figure 1: Fermat's spiral problem. (a) Samples belonging to two classes are distributed in a two-dimensional space, forming a spiral shape. A possible decision boundary is also plotted. If one walks from point A to B along the decision boundary, at any given point (say, point C), one obtain a liner problem locally. (b) By projecting the transformed data $\bar{\mathbf{z}}_n$ onto the direction specified by $\mathbf{w}$, most samples have positive margins.

After the margins are defined, the problem of learning feature weights can be directly solved within a margin framework. Two most popular margin formulations are SVM [14] and logistic regression [16]. Due to the nonnegative constraint on $\mathbf{w}$, the SVM formulation represents a large-scale optimization problem, while the problem size can not be reduced by transforming it into the dual domain. For computational convenience, we therefore perform the estimation in the logistic regression formulation [16]. In applications with a huge amount of features (e.g., molecular classification), we expect that most of features are irrelevant. To encourage the sparseness, one commonly used strategy is to add ℓ_1 penalty of $\mathbf{w}$ to an objective function. Examples, where this strategy has been shown successful, include LASSO for regression [17], ℓ_1-SVM [11] and ℓ_1 regularized logistic regression [12] for classification. Accomplishing sparse solutions by introducing the ℓ_1 penalty has been theoretically justified (see, for example, [18] and the references therein). With the ℓ_1 penalty, we obtain the following optimization problem:

$$\text{(2.6)} \qquad \begin{array}{rl} \min\limits_{\mathbf{w}} & \sum\limits_{n=1}^{N} \log\left(1+\exp(-\mathbf{w}^T\bar{\mathbf{z}}_n)\right) + \lambda\|\mathbf{w}\|_1 \\ \text{s.t.} & \mathbf{w} \geq 0 \ , \end{array}$$

where λ is a parameter that controls the penalty strength and consequently the sparseness of the solution. The optimization formulation (2.6) can also be written as:

$$\text{(2.7)} \qquad \begin{array}{rl} \min\limits_{\mathbf{w}} & \sum\limits_{n=1}^{N} \log\left(1+\exp(-\mathbf{w}^T\bar{\mathbf{z}}_n)\right) \\ \text{s.t.} & \|\mathbf{w}\|_1 \leq \beta, \mathbf{w} \geq 0 \ . \end{array}$$

For every solution $\mathbf{w}$ to (2.6) found using some particular value of λ, there is a corresponding value of β in (2.7) that will give the same solution. The optimization problem of (2.7) has an interesting interpretation. If we adopt a classification rule where $\mathbf{x}_n$ is correctly classified if and only if margin $\bar{\rho}_n(\mathbf{w}) \geq 0$, then $\sum_{n=1}^{N} I(\bar{\rho}_n(\mathbf{w}) < 0)$ is the leave-one-out (LOO) classification error induced by using $\mathbf{w}$, where $I(\cdot)$ is the indicator function. Since the logistic loss function is an upper bound of the misclassification loss function, up to a difference of a constant factor, (2.7) can be interpreted as finding a weighted feature space so that the upper bound of the LOO classification error in the induced space is minimized. Hence, the algorithm has two levels of regularization, i.e., the implicit LOO and explicit ℓ_1 regularization. We will shortly see that this property, together with the convergence property, leads to superior performance of the algorithm in the presence of copious irrelevant features, and that due to the LOO regularization, the performance of the algorithm is largely insensitive to a specific choice of λ.

Since $\bar{\mathbf{z}}_n$ implicitly depends on $\mathbf{w}$ through the probabilities $P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})$ and $P(\mathbf{x}_i = \text{NM}(\mathbf{x}_n)|\mathbf{w})$, we use a fixed-point recursion method to solve for $\mathbf{w}$. In each iteration, $\bar{\mathbf{z}}_n$ is first computed by using the previous estimate of $\mathbf{w}$, which is then updated by solving the optimization problem of (2.6). The iterations are carried out until convergence. It is interesting to note that though local learning is a highly nonlinear process, in each iteration we deal with a linear model.

For fixed $\bar{\mathbf{z}}_n$, (2.6) is a constrained convex optimization problem. Due to the nonnegative constraint on $\mathbf{w}$, it cannot be solved directly by using a gradient descent method. To overcome this difficulty, we reformulate the problem slightly as:

(2.8)

$$\min_{\mathbf{v}} \sum_{n=1}^{N} \log\left(1+\exp\left(-\sum_{j} {v_j}^2 \bar{z}_n(j)\right)\right) + \lambda\|\mathbf{v}\|_2^2 \ ,$$

thus obtaining an unconstrained optimization problem. It is easy to show that at the optimum solution we have $w_j = {v_j}^2, 1 \leq j \leq J$. The solution of $\mathbf{v}$ can be readily found through gradient descent with a simple update rule:

(2.9)

$$\mathbf{v} \leftarrow \mathbf{v} - \eta\left(\lambda\mathbf{1} - \sum_{n=1}^{N} \frac{\exp(-\sum_j {v_j}^2 \bar{z}_n(j))}{1+\exp(-\sum_j {v_j}^2 \bar{z}_n(j))}\bar{\mathbf{z}}_n\right) \otimes \mathbf{v} \ ,$$

where $\otimes$ is the Hadamard operator, and η is the learning rate determined by the standard line search. Note that the objective function of (2.8) is no longer a convex function, and thus a gradient descent method may find a local minimizer or a saddle point. The following theorem shows that if the initial point is properly selected, the solution obtained when the gradient vanishes is the global minimizer.

THEOREM 2.1. *Let $f(\mathbf{x})$ be a strictly convex function of $\mathbf{x} \in \mathbb{R}^J$ and $g(\mathbf{x}) = f(\mathbf{y})$, where $\mathbf{y} = [y_1, \cdots, y_J]^T = [x_1^2, \cdots, x_J^2]^T$. If $\frac{\partial g}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}^+} = \mathbf{0}$, then $\mathbf{x}^+$ is not a local minimizer, but a saddle point or a global minimizer of $g(\mathbf{x})$. Moreover, if $\mathbf{x}^+$ is found through gradient descent with an initial point $x_j^{(0)} \neq 0, 1 \leq j \leq J$, then $\mathbf{x}^+$ is the global minimizer of $g(\mathbf{x})$.*

Proof. Due to the space limitation, we only present the sketch of the proof. The proof is done by examining the properties of Hessian matrices. It can be proved that a given stationary point $\mathbf{x}^+$ is a global minimizer of $g(\mathbf{x})$ if Hessian matrix $\mathbf{H}(\mathbf{x}^+)$ is positive semidefinite, and a saddle point otherwise. If stationary point $\mathbf{x}^+$ is found with an initial point $x_j^{(0)} \neq 0, 1 \leq j \leq J$, then $\mathbf{x}^+$ is the global minimizer of $g(\mathbf{x})$ since the saddle points are not reachable via gradient descent.

For fixed $\bar{\mathbf{z}}_n$, the objective function of (2.6) is a strictly convex function of $\mathbf{w}$. Theorem 2.1 assures that in each iteration, via gradient descent, reaching the global optimum solution of $\mathbf{w}$ is guaranteed. After the feature weighting vector is found, the pairwise distances among data samples are re-evaluated using the updated feature weights, and the probabilities $P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w})$ and $P(\mathbf{x}_j{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})$ are re-computed using the newly obtained pairwise distances. The

```
Feature Selection Algorithm

Input: Data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, stop criterion $\theta$, kernel width $\sigma$, regularization parameters $\lambda$;
(1) Initialization: set $\mathbf{w}^{(0)} = \mathbf{1}, t = 0$;
(2) Repeat
  (3) Compute $d(\mathbf{x}_n, \mathbf{x}_i|\mathbf{w}^{(t)}), \forall \mathbf{x}_n, \mathbf{x}_i \in \mathcal{D}$;
  (4) Calculate $P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w}^{(t)})$ and
      $P(\mathbf{x}_j = \text{NH}(\mathbf{x}_n)|\mathbf{w}^{(t)})$ as in (2.4) and (2.5);
  (5) Solve for $\mathbf{v}$ as in (2.9);
  (6) $w_j^{(t+1)} = (v_j)^2, 1 \leq j \leq J$;
  (7) t=t+1;
(8) Until $\|\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)}\| < \theta$
Output: $\mathbf{w} = \mathbf{w}^{(t)}$.
```

Figure 2: Pseudo-code of the proposed algorithm.

two steps are iterated until convergence. The implementation of the algorithm is very simple. It is coded in Matlab with less than one hundred lines. Except for the standard linear search, no other built-in Matlab functions are used. The pseudo-code of the algorithm is presented in Fig. 2.

2.1 Convergence Analysis This section presents the convergence analysis of our algorithm. We begin by studying its asymptotic behavior. If $\sigma \to +\infty$, for all $\mathbf{w} \geq 0$ and $i \in \mathcal{M}_n$, we have $\lim\limits_{\sigma\to+\infty} P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w}) = \dfrac{1}{|\mathcal{M}_n|}$, since $\lim\limits_{\sigma\to+\infty} k(d) = 1$. On the other hand, if $\sigma \to 0$, by assuming that for all $\mathbf{x}_n$, $d(\mathbf{x}_n, \mathbf{x}_i|\mathbf{w}) \neq d(\mathbf{x}_n, \mathbf{x}_j|\mathbf{w})$ if $i \neq j$, we have $\lim\limits_{\sigma\to 0} P(\mathbf{x}_i = \text{NM}(\mathbf{x}_n)|\mathbf{w}) = 1$ if $d(\mathbf{x}_n, \mathbf{x}_i|\mathbf{w}) = \min\limits_{j\in\mathcal{M}_n} d(\mathbf{x}_n, \mathbf{x}_j|\mathbf{w})$ and 0 otherwise. Similar asymptotic behavior holds for $P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})$. From the above analysis, it follows that for $\sigma \to +\infty$ the algorithm converges to a unique solution in one iteration, since $P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w})$ and $P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})$ are constants for any initial feature weights. On the other hand, for $\sigma \to 0$, rarely do we empirically observe that the algorithm converges. This suggests that the convergence behavior and convergence rate of the algorithm are fully controlled by the kernel width, which is formally stated in the following theorem.

THEOREM 2.2. *For the proposed feature selection algorithm, there exists σ^* such that $\lim_{t\to+\infty} \|\mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}\| = 0$ whenever $\sigma > \sigma^*$. Moreover, for a fixed $\sigma > \sigma^*$, the algorithm converges to a unique solution for any nonnegative initial feature weights $\mathbf{w}^{(0)}$.*

We use the Banach fixed point theorem to prove the convergence theorem. We first state the fixed point theorem without proof, which can be found for example in [19].

DEFINITION 2.1. *Let $\mathcal{U}$ be a subset of a normed space $\mathcal{Z}$, and $\|\cdot\|$ is a norm defined in $\mathcal{Z}$. An operator $T : \mathcal{U} \to \mathcal{Z}$ is called a contraction operator if there exists a constant $q \in [0, 1)$ such that $\|T(x) - T(y)\| \leq q\|x - y\|$ for every $x, y \in \mathcal{U}$. q is called the contraction number of T.*

DEFINITION 2.2. *An element of a normed space $\mathcal{Z}$ is called a fixed point of $T : \mathcal{U} \to \mathcal{Z}$ if $T(x) = x$.*

THEOREM 2.3. (FIXED POINT THEOREM) *Let T be a contraction operator mapping a complete subset $\mathcal{U}$ of a normed space $\mathcal{Z}$ into itself. Then the sequence generated as $x^{(t+1)} = T(x^{(t)})$, $\quad t = 0, 1, 2, \cdots$, with arbitrary $x^{(0)} \in \mathcal{U}$, converges to the unique fixed point x^* of T. Moreover, the following estimation error bounds hold:*

$$\begin{aligned} &\|x^{(t)} - x^*\| \leq \tfrac{q^t}{1-q}\|x^{(1)} - x^{(0)}\| , \\ \textit{and} \quad &\|x^{(t)} - x^*\| \leq \tfrac{q}{1-q}\|x^{(t)} - x^{(t-1)}\| . \end{aligned} \tag{2.10}$$

Proof. (of Theorem 2.2): The gist of the proof is to identify a contraction operator for the algorithm, and make sure that the conditions of Theorem 2.3 are met. To this end, we define $\mathcal{P} = \{\mathbf{p} : \mathbf{p} = [P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w}), P(\mathbf{x}_j{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})]\}$, and specify the first step of the algorithm in a functional form as $T1 : \mathbb{R}^J_{\geq 0} \to \mathcal{P}$, where $T1(\mathbf{w}) = \mathbf{p}$, and the second step as $T2 : \mathcal{P} \to \mathbb{R}^J_{\geq 0}$, where $T2(\mathbf{p}) = \mathbf{w}$. Here $\mathbb{R}^J_{\geq 0} = \{\mathbf{w} : \mathbf{w} \in \mathbb{R}^J, \mathbf{w} \geq 0\}$. Then, the algorithm can be written as $\mathbf{w}^{(t)} = (T2 \circ T1)(\mathbf{w}^{(t-1)}) \triangleq T(\mathbf{w}^{(t-1)})$, where $(\circ)$ denotes functional composition and $T : \mathbb{R}^J_{\geq 0} \to \mathbb{R}^J_{\geq 0}$. Since $\mathbb{R}^J_{\geq 0}$ is a closed subset of normed space $\mathbb{R}^J$ and complete, T is an operator mapping complete subset $\mathbb{R}^J_{\geq 0}$ into itself. Next, note that for $\sigma \to +\infty$, the algorithm converges with one step. We have $\lim\limits_{\sigma\to+\infty} \|T(\mathbf{w}_1, \sigma) - T(\mathbf{w}_2, \sigma)\| = 0$, for every $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^J_{\geq 0}$. Therefore, in the limit, T is a contraction operator with contraction constant $q = 0$, that is, $\lim\limits_{\sigma\to+\infty} q(\sigma) = 0$. Therefore, for every $\varepsilon > 0$, there exists σ^* such that $q(\sigma) \leq \varepsilon$ whenever $\sigma > \sigma^*$. By setting $\varepsilon < 1$, the resulting operator T is a contraction operator. By the Banach fixed point theorem, our algorithm converges to a unique fixed point provided the kernel width is properly selected. The above arguments establish the convergence theorem of the algorithm.

The theorem ensures the convergence of the algorithm if the kernel width is properly selected. This is a very loose condition, as our empirical results show that the algorithm always converges for a sufficiently large kernel width. Also, the error bound in (2.10) tells us that the smaller the contraction number, the tighter the error bound and hence the faster the convergence rate. Our experiments suggest that a larger kernel width yields a faster convergence. Unlike many other machine learning algorithms (e.g., neural networks), the convergence and the solution of our algorithm are not affected by the initial value if the kernel width is fixed. Even if the initial feature weights were wrongly selected (e.g., investigators have no or false prior), and the algorithm started computing erroneous nearest misses and hits for each sample, the theorem assures that the algorithm will eventually converge to a unique solution. This property is experimentally demonstrated in Sec. 3.1 (see Fig. 4).

2.2 Computational Complexity and Fast Implementation The main complexity of the algorithm comes from computing pairwise distances between data samples. Thus, the computational complexity of the algorithm is $\mathcal{O}(TN^2J)$, where T is the number of iterations, J is the feature dimensionality, and N is the number of data samples. We empirically find that our algorithm usually converges within a few iterations (see Fig. 3). Hence, its computational complexity is comparable to that of RELIEF. A close look at the update equation of $\mathbf{v}$, given by (2.9), allows us to further reduce complexity. If some elements of $\mathbf{v}$ are very close to zero (say less than 10^{-4}), the corresponding features can be eliminated from further consideration with a negligible impact on the subsequent iterations, thus providing a built-in mechanism for automatically removing irrelevant features during learning.

2.3 Feature Selection for Multiclass Problems Some existing feature selection algorithms, originally designed for binary problems, can be naturally extended to multiclass settings, while for others the extension is not straightforward. In particular, for both embedded and wrapper methods, the extension largely depends on the capability of a classifier to handle multiclass problems. In many cases, a multiclass problem is first decomposed into several binary ones, and then feature selection is performed for each binary problem. This strategy further increases the computational burden of embedded and wrapper methods. Our algorithm does not suffer from this problem. A natural extension of the margin defined in (2.1) to multiclass problems is [20]:

(2.11)
$$\begin{aligned} &\rho_n(\mathbf{w}) \\ =\ &\min_{\{c\in\mathcal{Y}, c\neq y_n\}} d(\mathbf{x}_n, \mathrm{NM}^{(c)}(\mathbf{x}_n)|\mathbf{w}) - d(\mathbf{x}_n, \mathrm{NH}(\mathbf{x}_n)|\mathbf{w})\ , \\ =\ &\min_{\mathbf{x}_i\in\mathcal{D}\backslash\mathcal{D}_{y_n}} d(\mathbf{x}_n, \mathbf{x}_i|\mathbf{w}) - d(\mathbf{x}_n, \mathrm{NH}_n|\mathbf{w})\ , \end{aligned}$$

where $\mathcal{Y}$ is the set of class labels, $\mathrm{NM}^{(c)}(\mathbf{x}_n)$ is the nearest neighbor of $\mathbf{x}_n$ from class c, and $\mathcal{D}_c$ is a subset of $\mathcal{D}$ containing only samples from class c. The derivation of our feature selection algorithm for multiclass problems by using the margin defined in (2.11) is straightforward.

3 Experiments

The effectiveness of the proposed algorithm is empirically validated on eight synthetic and real-world datasets. We demonstrate that our algorithm is capable of handling problems with an extremely large feature dimensionality.

3.1 Spiral Problem We perform a simulation study on an artificially generated dataset, carefully designed to verify various important properties of the algorithm. This example, also called Fermat's spiral problem, is a binary classification problem. Each class has 230 samples distributed in a two-dimensional space, forming a spiral shape, as illustrated in Fig. 1. In addition to the first two relevant features, each sample is represented by a varying number of irrelevant features, where this number is set to $\{50, 100, 500, 1000, 5000, 10000, 20000, 30000\}$. The number 30000 exceeds by far the amount of features experienced in many scientific fields. For example, human beings have about 25000 genes, and hence nearly all gene expression microarray platforms have less than 25000 probes. The added irrelevant features are independently sampled from the zero-mean and unit-variance Gaussian distribution. Our task is to identify the first two relevant features. Note that only if these two features are used simultaneously can the two classes of samples be well separated. For most existing feature selection algorithms, identifying the two relevant features in this example is an extremely challenging task.

Fig. 3 illustrates the dynamics of our algorithm performed on the spiral data with 10000 irrelevant features. The algorithm iteratively refines the estimates of weight vector $\mathbf{w}$ and probabilities $P(\mathbf{x}_i{=}\mathrm{NH}(\mathbf{x}_n)|\mathbf{w})$ and $P(\mathbf{x}_i{=}\mathrm{NM}(\mathbf{x}_n)|\mathbf{w})$ until convergence. Each sample is colored according to its probability of being the nearest miss or hit of a given sample indicated by a black cross. We observe that, with a uniform initial feature weights, the nearest neighbors defined in the original feature space can be completely different from the true ones. The plot also shows that the algorithm converges to a perfect solution in just three iterations.

Fig. 4 presents the feature weights that our algorithm learns on the spiral data for a varying number of irrelevant features. The results are obtained for parameters σ and λ respectively set to 2 and 1, while the same solution holds for a wide range of other values of kernel widths and regularization parameters (insensitivity to a specific choice of these parameters will be discussed shortly). Fig. 4 shows that our algorithm performs remarkably well over a wide range of feature-dimensionality values that are of practical interest, with the *same* parameters. We also note that the feature weights learned across all feature-dimensionality values are almost identical. This result is a consequence of Theorem 2.2, which can be explained as follows. Suppose that we have two spiral datasets having 5000 and 10000

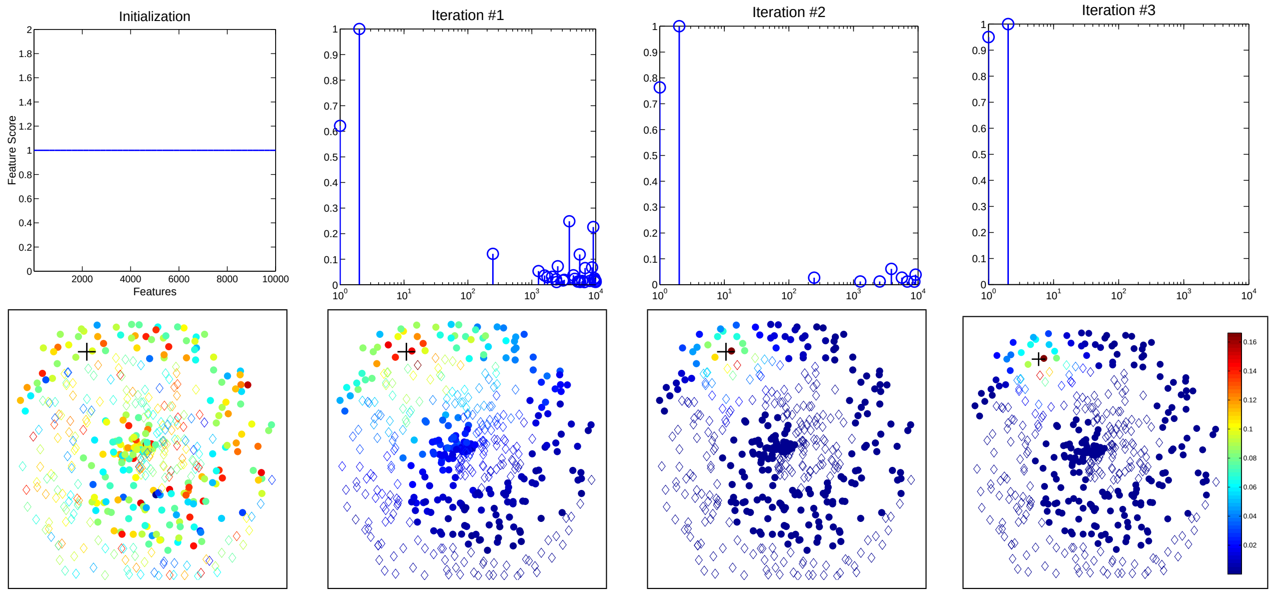


Figure 3: The algorithm iteratively refines the estimates of weight vector $\mathbf{w}$ and probabilities $P(\mathbf{x}_i{=}\text{NH}(\mathbf{x}_n)|\mathbf{w})$ and $P(\mathbf{x}_i{=}\text{NM}(\mathbf{x}_n)|\mathbf{w})$ until convergence. The result is obtained on the spiral data with 10000 irrelevant features. Each sample is colored according to its probability of being the nearest miss or hit of a given sample indicated by a black cross. The plot shows that the algorithm converges to a perfect solution in just three iterations.

irrelevant features, respectively. Also, suppose that we first perform the algorithm on the second dataset, and that after some iterations the algorithm finds 5000 irrelevant features whose weights are very close to zero. Then, both problems are almost identical, except that the first problem has a uniform initial point (see line 1 of Fig. 2) and the second problem has a non-uniform initial point. By Theorem 2.2, the algorithm converges to the same solution for both problems. Of course, due to the randomness of irrelevant features and the finite number of iteration steps, the two solutions are slightly different. For example, in Fig. 4, for the dataset with 30000 features, the algorithm selects one false feature as relevant, in addition to the two relevant ones. However, the weight associated with the selected irrelevant feature is much smaller than the weights of the two relevant ones.

One may be interested to know how many irrelevant features should be added to the dataset before the algorithm fails. We conduct an experiment where the number of irrelevant features are continuously increased. We find that the algorithm attains the almost identical solutions to those presented in Fig. 4 until 100000 irrelevant features are injected into the dataset. The algorithm fails simply because the computer is out of memory. This result suggests that our algorithm is capable of handling an extremely large number of irrelevant features, far beyond that needed in most data-analysis settings one may currently encounter. This result is very encouraging, and deserves further theoretical analyses.

Our algorithm is computationally very efficient. Fig. 5 shows the CPU time it takes the algorithm to perform feature selection on the spiral dataset with different numbers of irrelevant features. The computer setting is Pentium4 2.80GHz with 2.00GB RAM. The stopping criterion is $\theta = 0.01$. As can be seen, the algorithm runs for only $3.5s$ for the problem with 100 features, $37s$ for 1000 features, and $372s$ for 20000 features. The computational complexity is linear with respect to the feature dimensionality. To the best of our knowledge, no wrapper method has computational complexity even close to ours. Depending on the classifier used in search for relevant features, it may take several hours for a wrapper method to analyze the same dataset with only 1000 features, and yet there is no guarantee that the optimal solution will be reached, due to heuristic search.

The kernel width σ and the regularization parameter λ are two input parameters of the algorithm. Alternatively, they can be estimated through cross validation on training data. It is well-known that cross validation may produce an estimate with a large variance. Fortunately, this does not pose a serious concern for our algorithm. In Figs. 6 and 7, we plot the feature weights learned with different kernel widths and regularization parameters. The algorithm performs well over a wide range of parameter values, yielding always the largest weights for the first two relevant features, while the other weights are significantly smaller. This suggests that the algorithm's performance is largely insensitive to a specific choice of parameters σ and λ, which makes parameter tuning

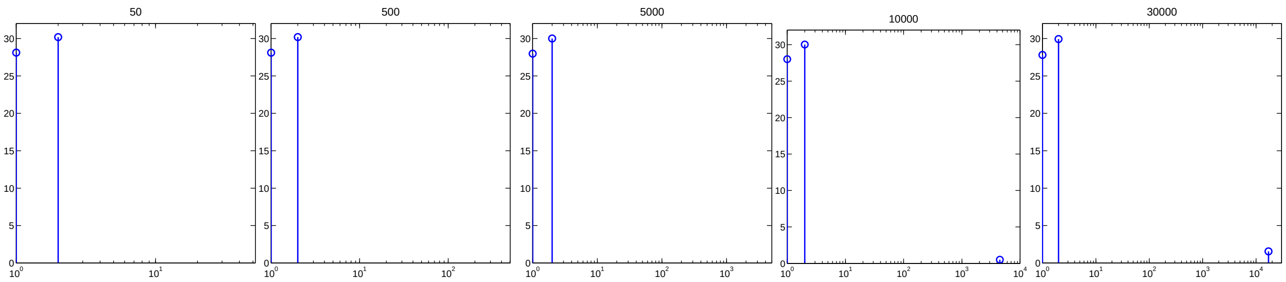


Figure 4: Feature weights learned on the spiral dataset with different numbers of irrelevant features, ranging from 50 to 30000. The y-axis represents the values of feature weights, and the x-axis is the number of features, where the first two are always fixed to represent the two relevant features. Zero-valued feature weights indicate that the corresponding features are not relevant. The feature weights learned across all dimensionality are almost identical, for the same input parameters.

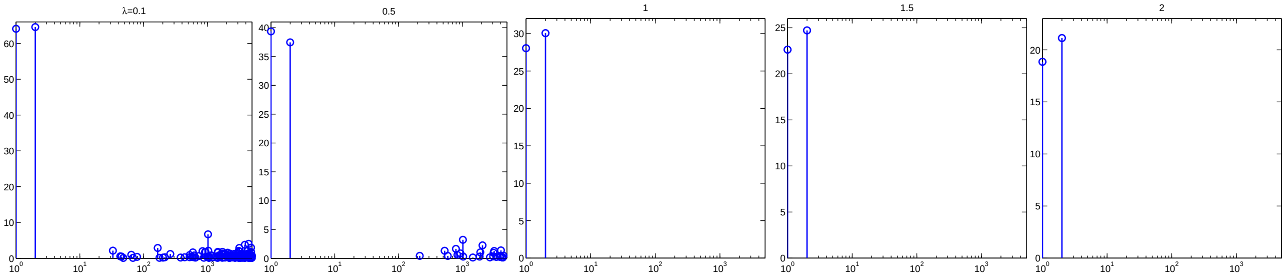


Figure 6: Feature weights learned on the spiral dataset with 5000 irrelevant features, for a fixed kernel width $\sigma = 2$, and different regularization parameters $\lambda \in \{0.1, 0.5, 1, 1.5, 2\}$.

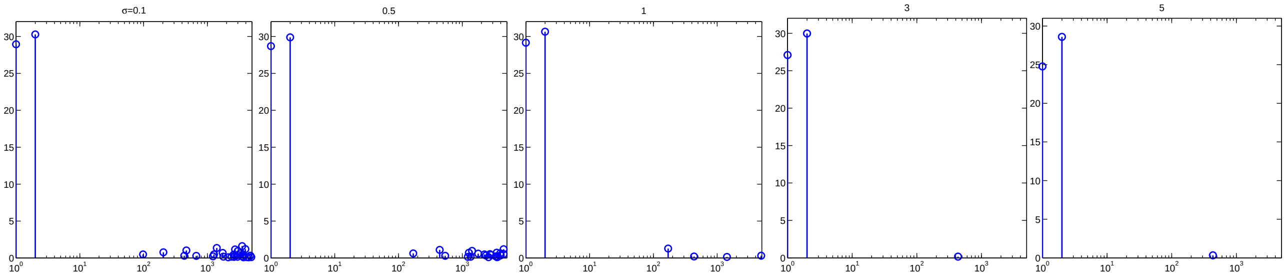


Figure 7: Feature weights learned on the spiral dataset with 5000 irrelevant features, for a fixed regularization parameter $\lambda = 1$, and different kernel widths $\sigma \in \{0.1, 0.5, 1, 3, 5\}$.

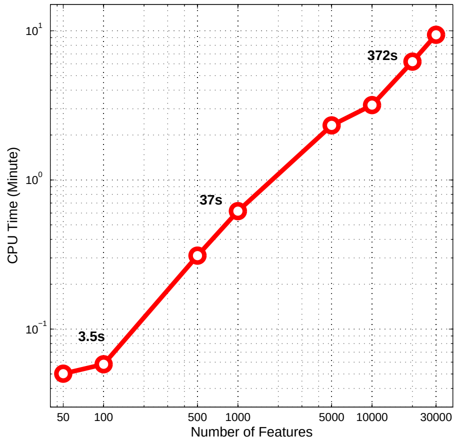


Figure 5: The run-times of our algorithm on the spiral dataset with a varying number of irrelevant features, ranging from 50 to 30000. The plot demonstrates linear complexity with respect to the feature dimensionality.

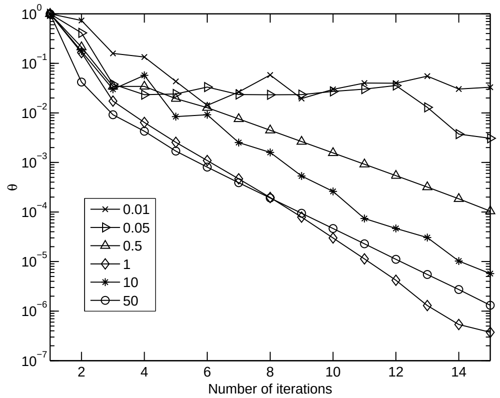


Figure 8: Convergence analysis of our algorithm, performing on the spiral dataset with 5000 irrelevant features, for $\lambda = 1$ and $\sigma \in \{0.01, 0.05, 0.5, 1, 10, 50\}$. The plots present $\theta = \|\mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}\|_2$ as a function of the number of iteration steps.

Table 1: Summary of UCI datasets. The number of artificial, irrelevant features added to the original ones is indicated in the parentheses.

Dataset	**Train**	**Test**	**Feature**
twonorm	400	7000	20(5000)
waveform	400	4600	21(5000)
banana	468	300	2(5000)
thyroid	140	75	5(5000)
diabetics	468	300	8(5000)
heart	170	100	13(5000)
splice	400	2175	60(5000)

and hence the implementation of our algorithm easy, even for researchers outside of the machine learning community.

Fig. 8 presents the convergence analysis of our algorithm on the spiral dataset with 5000 irrelevant features, for $\lambda = 1$ and different kernel widths $\sigma \in \{0.01, 0.05, 0.5, 1, 10, 50\}$. We observe that the algorithm converges for a wide range of σ values, and that in general a larger kernel width yields a faster convergence. These results validate our theoretical convergence analysis, presented in Sec. 2.1.

3.2 Experiments on UCI Datasets This section presents our feature selection results on seven benchmark UCI datasets [21]. The data information is summarized in Table 1. For each dataset, the set of original features is augmented by 5000 artificially generated irrelevant features. The irrelevant features are independently sampled from a Gaussian distribution with zero-mean and unit-variance. It should be noted that some features in the original feature sets may be irrelevant or weakly relevant, and hence may receive zero weights in our algorithm. Since the true relevance of the original features is unknown, to verify that our algorithm does indeed discover all relevant features, we compare the classification performance of SVM (with the RBF kernel) in two cases: (1) when only the original features are used (i.e., without 5000 useless features), and (2) when the features selected by our algorithm are used. It is well known that SVM is very robust against noise, and that the presence of a few irrelevant features in the original feature sets should not significantly affect its performance. Hence, the classification performance of SVM obtained in the first case should be very close to that of SVM performed on the optimum feature subsets that are unknown to us a priori. Essentially, we are comparing our algorithm with an optimal feature selection algorithm. If SVM performs similarly in both cases, we may conclude that our algorithm achieves close-to-optimum solutions.

The structural parameters of SVM are estimated through ten-fold cross validation on training data. To further demonstrate that the performance of our algorithm is largely insensitive to a specific choice of the input parameters, we use kernel width $\sigma = 2$ and regularization parameter $\lambda = 1$ for all seven UCI datasets. The stopping criterion is $\theta = 0.01$. The algorithm is run 10 times for each dataset. In each run, a dataset is randomly partitioned into training and test sets. The averaged classification errors and standard deviations of SVM are reported in Table 2. The false discovery rate (FDR), defined as the ratio between the number of artificially added, irrelevant features with non-zero weights and the total number of irrelevant features (i.e., 5000), is reported in Table 2. Feature weights, learned in one sample trial, for each of seven datasets, are shown in Fig. 9. From these experi-

Table 2: Classification errors on test data and their standard deviations ($\%$) of SVM performed on the seven UCI datasets contaminated by 5000 useless features. The second column records the performance of SVM performed on the features selected by our algorithm, and the third column is the results of SVM obtained by using the original features. The last two columns present the false discovery rates (FDR) and CPU times, and the last row presents the averaged FDR and CPU time.

	SVM	**SVM**	**FDR**	**CPU time**
Dataset	(selected features)	(original features)		(second per run)
twonorm	2.6(0.2)	2.6(0.2)	0/1000	162
waveform	11.7(0.9)	10.1(0.6)	1.6/1000	157
banana	10.9(0.5)	10.9(0.5)	0.1/1000	97
diabetics	23.7(1.1)	24.9(1.3)	1.8/1000	267
thyroid	5.3(2.4)	4.7(2.1)	0.1/1000	13
heart	17.2(4.2)	18.4(4.0)	0.4/1000	73
splice	12.9(2.1)	14.4(0.9)	1.0/1000	330
Average	/	/	0.7/1000	157

mental results, we observe the following:

(1) From Table 2, we observe that SVM using the features identified by our algorithm performs similarly or even slightly better than SVM using the original features. It should be emphasized that the results are obtained in the presence of a huge number of irrelevant features, without estimating the optimal input parameters.

(2) In addition to successfully identifying relevant features, our algorithm performs remarkably well in removing irrelevant ones. The false discovery rate, averaged over seven datasets, is only $0.7/1000$. The feature weights learned on one realization of each dataset are plotted in Fig. 9. For ease of presentation, the maximum value of each feature weight vector is normalized to 1. For some datasets (e.g., *banana* and *thyroid*), the weights of the false positives are very small, and we experimentally find that it is possible to achieve a perfect solution by slightly increasing the regularization parameter λ.

(3) The averaged CPU times of seven datasets are presented in Table 2. Our algorithm is capable of processing thousands of features within a few minutes on personal computers.

4 Conclusion

The newly proposed algorithm is formulated based on the concept that a given complex problem can be more easily, and yet accurately enough, analyzed by parsing it into a set of locally linear problems. Local learning allows us to capture local structure of the data, while the parameter estimation is performed globally to avoid possible overfitting. In comparison with other feature selection algorithms, our algorithm has many advantages. We have experimentally demonstrated that our algorithm is capable of handling problems with an extremely large feature dimensionality, without making any assumption on the underlying data distributions. Although, for clarity, we have mainly considered only binary problems, we have shown that the extension of our algorithm to multiclass settings is straightforward. Unlike most prior work whose implementation demands an expertise in machine learning, ours is very easy to implement. As with most machine learning algorithms, ours has two user defined parameters that can alternatively be estimated through cross validation. We have experimentally shown that the performance of our algorithm is largely insensitive to a wide range of values for these two parameters, which makes parameter tuning and hence the implementation of our algorithm easy in real applications.

Due to its high accuracy, low complexity, and simple implementation, we believe that the proposed framework may hold major potential for eventually solving the feature selection problem. Even in the current formulation, our algorithm is already capable of handling most feature selection problems one encounters in scientific research. Considering the increased demand for analyzing data with a large number of features in many research fields, we expect that the work presented in this paper will make a broad impact.

References

[1] R. Kohavi and G. H. John, *Wrappers for feature subset selection*, Artifi. Intell., 97 (1997), pp. 273–324.

[2] P. Pudil, J. Novovicova, *N*ovel methods for subset selection with respect to problem knowledge, *IEEE Intell. Syst.*, 13 (1998), pp. 66–74.

[3] T. G. Dietterich and G. Bakiri, *Solving multiclass learning problems via error-correcting output codes*, J. Artif. Intell. Res., 2 (1995), pp. 263–286.

[4] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff, *Embedded methods*, in Feature Extraction, Foundations and Applications, Springer-Verlag, 2006.

[5] K. Kira and L. A. Rendell, *A practical approach to feature*

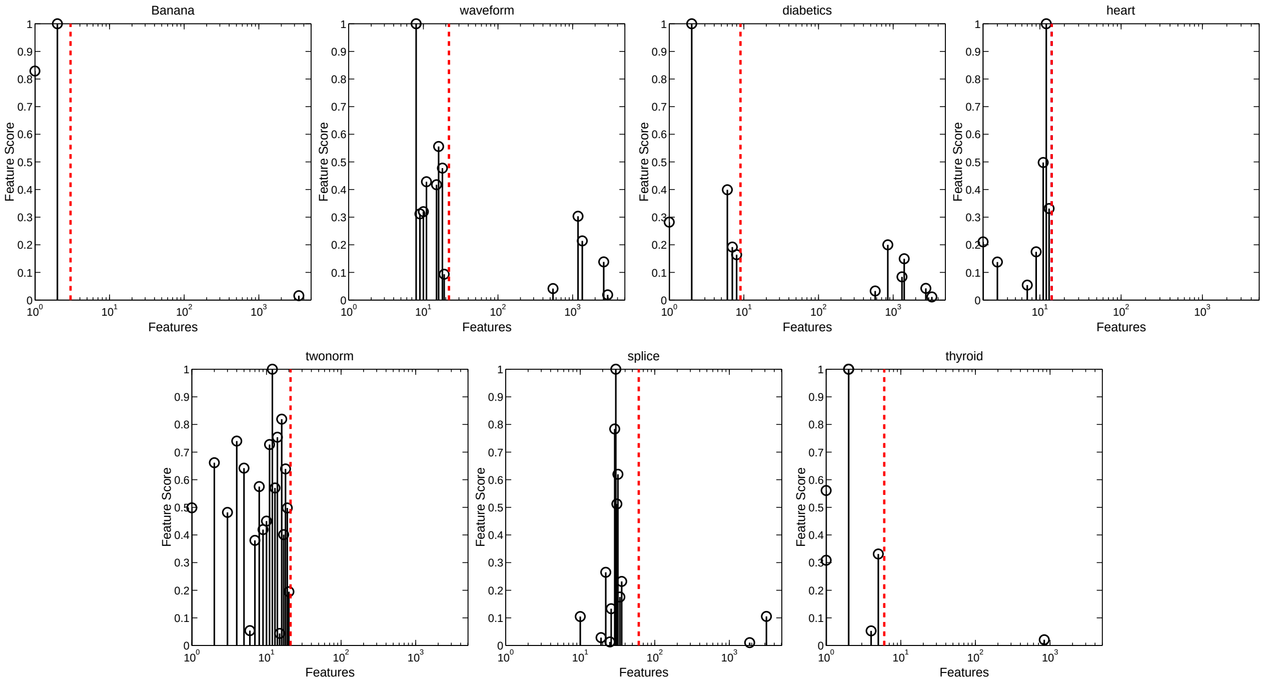


Figure 9: Feature weights learned in one sample trial on seven UCI datasets. The dashed line indicates the number of original features. The weights plotted on the left side of the dashed line are associated with the original features, while those on the right, with the additional 5000 irrelevant features. Some of the original features are found irrelevant.

selection, Proc. 9th Int. Conf. Mach. Learn., (1992), pp. 249–256.

[6] I. Kononenko, *Estimating attributes: analysis and extensions of RELIEF*, Eur. Conf. Mach. Learn., (1994), pp. 171–182.

[7] Y. Sun and J. Li, *Iterative RELIEF for feature weighting, Proc. 21st Int. Conf. Mach. Learn.*, (2006), pp. 913–920.

[8] R. Gilad-Bachrach, A. Navot, and N. Tishby, *Margin based feature selection-theory and algorithms*, Proc. 19th Int. Conf. Mach. Learn., (2004), pp. 43–50.

[9] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, *Feature selection for SVMs*, Advances in Neural Information Processing Systems, (2001), pp. 668–674.

[10] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, *Gene selection for cancer classification using support vector machines, Mach. Learn.*, 46 (2002), pp. 389-422.

[11] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, *1-norm support vector machines*, Advances in Neural Information Processing Systems, 16 (2004).

[12] A. Y. Ng, *Feature selection, L1 vs. L2 regularization, and rotational invariance, Proc. 21st Int. Conf. Mach. Learn.*, 69 (2004), pp. 78–86.

[13] A. Y. Ng and M. I. Jordan, *Convergence rates of the Voting Gibbs classifier, with application to Bayesian feature selection, Proc. 18th Int. Conf. Mach. Learn.*, (2001), pp. 377–384.

[14] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[15] A. Dempster, N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm (with discussion)*, J. R. Stat. Soc. Ser. B, 39 (1977), pp. 1–38.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[17] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. R. Stat. Soc. Ser. B, 58 (1996), pp. 267–288.

[18] D. Donoho, M. Elad, *Optimally sparse representations in general nonorthogonal dictionaries by $l1$ minimization*, Proc. Natl. Acad. Sci. U.S.A., 100 (2003), pp. 2197–2202.

[19] R. Kress, *Numerical Analysis*, Springer-Verlag, New York, 1998.

[20] Y. Sun, S. Todorovic, J. Li, and D. Wu, *Unifying error-correcting and output-code AdaBoost through the margin concept*, Proc. 22nd Int. Conf. Mach. Learn., (2005), pp. 872–879.

[21] C. Blake and C. Merz, UCI repository of machine learning databases, 1998.