International Journal of Foundations of Computer Science
Vol. 26, No. 3 (2015) 321–341

DOI: 10.1142/S0129054115500185

World Scientific
www.worldscientific.com

Novel Randomized Feature Selection Algorithms

Subrata Saha* and Sanguthevar Rajasekaran†

Department of Computer Science and Engineering
University of Connecticut, Storrs, CT, USA
**subrata.saha@engr.uconn.edu*
†*rajasek@engr.uconn.edu*

Rampi Ramprasad

Department of Materials Science and Engineering
University of Connecticut, Storrs, CT, USA
rampi@ims.uconn.edu

Received 11 June 2014
Accepted 3 November 2014
Communicated by Sartaj Sahni

Feature selection is the problem of identifying a subset of the most relevant features in the context of model construction. This problem has been well studied and plays a vital role in machine learning. In this paper we present three randomized algorithms for feature selection. They are generic in nature and can be applied for any learning algorithm. Proposed algorithms can be thought of as a random walk in the space of all possible subsets of the features. We demonstrate the generality of our approaches using three different applications. The simulation results show that our feature selection algorithms outperforms some of the best known algorithms existing in the current literature.

Keywords: Feature selection (FS); machine learning; data integrator (DI); gene selection algorithm (GSA); kernel ridge regression (KRR); sequential forward search (SFS).

1. Introduction

Feature Selection is defined as the process of selecting a subset of the most relevant features from a set of features. It involves discarding irrelevant, redundant and noisy features. Feature selection is also known as variable selection, attribute selection or variable subset selection in the fields of machine learning and statistics. The concept of feature selection is different from feature extraction. Feature extraction creates new features from the set of original features by employing a variety of methods such as linear combinations of features, projection of features from the original space into a transformed space, etc. We can summarize the usefulness of feature selection as follows: (1) Shorter training times: When irrelevant and redundant features are eliminated, the learning time decreases; (2) Improved model creation: The model built is more accurate and efficient; and (3) Enhanced generalization: It produces simpler and more generalized models.

A generic feature selection algorithm employs the following steps: (1) Select a subset of features; (2) Evaluate the selected subset; and (3) If the stopping condition is met then terminate else repeat Steps 1 and 2. The algorithm generates candidate subsets using different searching strategies depending on the application. Each of the candidate subsets is then evaluated based on an objective function. In the context of any learning algorithm, the objective function could be the accuracy. Note that for any learning algorithm there are two phases. In the first phase (known as the *training phase*), the learner is trained with a set of known examples. In the second phase (known as the *test phase*), the algorithm is tested on unknown examples. Accuracy refers to the fraction of test examples on which the learner is able to give correct answers. In the feature selection algorithm, if the current subset of features yields a better value for the objective function, the previous best solution is replaced with the current one. If not, the next candidate is generated. This process iterates over the search space until a stopping condition is satisfied. Finally, the best subset is validated by incorporating prior knowledge.

In this paper we introduce a variety of randomized techniques for feature selection. These techniques can be used in the context of any learning algorithm. Consider the space of all possible subsets of features. We start with a random subset s of the features and calculate its accuracy. We then choose a *random neighbor*[a] s' of s and compute its accuracy. If the accuracy of s' is greater than that of s, we move to the new subset s' and proceed with the search from this point. Otherwise, we proceed our search depending on different techniques proposed in our algorithms. Suppose if the accuracy of s' is smaller than that of s, we can stay with the subset s (with some probability p) or move to the subset s' with probability $1 - p$. We proceed with the search from the point we end up with. This process of searching the space is continued until no significant improvement in the accuracy can be obtained. Our randomized search techniques are generic in nature. We have employed it on three different applications and found that those are indeed scalable, reliable and efficient. Note that our algorithms resembles many local searching algorithms (such as Simulated Annealing (SA)). However, our algorithms are much simpler and differs from the others. For example, we do not employ the notion of *temperature* that SA utilizes.

The rest of this paper is organized as follows: Related works are summarized in Sec. 2. Some background information and preliminaries are presented in Sec. 3. In Sec. 4 we describe our proposed algorithms. Analyses of our algorithms such as time complexity and convergence proof are presented in Sec. 5. The performance of the algorithms and experimental results are presented in Sec. 6. Section 7 concludes the paper.

[a]The notion of a random neighbor is defined precisely in Sec. 4.

2. Related Works

In this section we provide a summary of some well-known feature selection algorithms. These algorithms differ in the way the candidate subsets are generated and in the evaluation criterion used. Some examples of feature extraction methods can be found in [34].

2.1. *Selection of candidate subset*

Subset selection begins with an initial subset that could be empty, the entire set of features, or some randomly chosen features. This initial subset can be changed in a number of ways. In forward selection strategy, features are added one at a time. In backward selection the least important feature is removed based on some evaluation criterion. Random search strategy randomly adds or removes features to avoid being trapped in a local maximum. If the total number of features is n, the total number of candidate subsets is 2^n. An exhaustive search strategy searches through all the 2^n feature subsets to find an optimal one. Clearly, this may not be feasible in practice [15]. A number of heuristic search strategies have been introduced to overcome this problem. The branch and bound method [22] exploits exhaustive search by maintaining and traversing a tree, but stops the search along a particular branch if a predefined boundary value is exceeded. The branch and bound method has been shown to be effective on many problem instances.

Greedy hill climbing strategies modify the current subset in such a way that results in the maximum improvement in the objective function (see e.g., [25]). Sequential forward search (SFS) [4, 5], sequential backward search (SBS), and bidirectional search [18] are some variations to the greedy hill climbing method. In these methods, the current subset is modified by adding or deleting features. SFS sequentially searches the feature space by starting from the empty set and selects the best single feature to add into the set in each iteration. On the contrary, SBS starts from the full feature set and removes the worst single feature from the set in each iteration. Both approaches add or remove features one at a time. Algorithms with sequential searches are fast and have a time complexity of $O(n^2)$. Sequential forward floating search (SFFS) and sequential backward floating search (SBFS) [23] combine the strategies followed by SFS and SBS. Some feature selection algorithms randomly pick subsets of features from the feature space by following some probabilistic steps and sampling procedures. Examples include evolutionary algorithms [8, 9], and simulated annealing [4]. The use of randomness helps in the avoidance of getting trapped in local maxima.

2.2. *Evaluation of the generated subset*

After selecting the subsets from the original set of features, they are evaluated using an objective function. One possible objective function is the accuracy of the predictive model. Feature selection algorithms can be broadly divided into two

categories: (1) wrapper, and (2) filter. In a wrapper method the classification or prediction accuracy of an inductive learning algorithm of interest is used for evaluation of the generated subset. For each generated feature subset, wrappers evaluate its accuracy by applying the learning algorithm using the features residing in the subset. Although it is a computationally expensive procedure, wrappers can find the subsets from the feature space with a high accuracy because the features match well with the learning algorithm. Filter methods are computationally more efficient than wrapper methods since they evaluate the accuracy of a subset of features using objective criteria that can be tested quickly. Common objective criteria include the mutual information, Pearson product-moment correlation coefficient, and the inter/intra class distance. Though filters are computationally more efficient than wrappers, often they produce a feature subset which is not matched with a specific type of predictive model and thus can yield worse prediction accuracies. Some of the filter-based methods are Chi-square [36], correlation-based (e.g., linear and rank), and entropy-based (e.g., information and gain-ratio) [37] filters.

3. Background Summary

We demonstrate the applicability of our proposed algorithms using three different applications. The applications of interest are: (1) the prediction of materials properties, (2) analysis of biological data, and (3) data integration. In this section we provide a brief summary on these applications.

3.1. *Materials property prediction*

If one wants to determine properties of a given unknown material, the traditional approaches are lab measurements or computationally intensive simulations (for example using the Density Functional Theory). An attractive alternative is to employ learning algorithms. The idea is to learn the desired properties from easily obtainable information about the material. In this paper we consider an infinite polymeric chain composed of XY_2 building blocks, with X = C, Si, Ge, or Sn, and Y = H, F, Cl, or Br. We are interested in estimating different properties of such chains including dielectric constant and band gap. We assume an infinite polymer chain with a repeat unit containing 4 distinct building blocks, with each of these 4 blocks being any of CH_2, SiF_2, $SiCl_2$, GeF_2, $GeCl_2$, SnF_2, or $SnCl_2$. By plotting the total dielectric constant (composed of the electronic and ionic contributions) and the electronic part of the dielectric constant against the computed band gap, we find some correlations between these three properties. While some correlations are self-evident (and expected) — such as the inverse relationship between the band gap and the electronic part of the dielectric constant, and the large dielectric constant of those systems that contain contiguous SnF_2 units — it is not immediately apparent if these observations may be formalized in order to allow for quantitative property predictions for systems (within this sub-class, of course) not originally considered.

For example, can we predict the properties of a chain with a repeat unit containing 8 building blocks (with each of the blocks being any of the aforementioned units)? In Sec. 5, we show that this can indeed be done with high-fidelity using our randomized search method.

We use specific sub-structures, or *motifs* or *scaffolds*, within the main structure to create the attribute vector. Let us illustrate this using the specific example of the polymeric dielectrics created using XY_2 building blocks. Say there are 7 possible choices (or motifs) for each XY_2 unit: CH_2, SiF_2, $SiCl_2$, GeF_2, $GeCl_2$, SnF_2, and $SnCl_2$. The attribute vector may be defined in terms of 6 fractions, $|f_1, f_2, f_3, f_4, f_5, f_6\rangle$, where f_i is the fraction of XY_2 type or motif i (note that $f_7 = 1 - \Sigma_{i=1}^{6} f_i$). One can extend the components of the attribute vector to include clusters of 2 or 3 XY_2 units of the same type occurring together; such an attribute vector could be represented as $|f_1, \ldots, f_6, g_1, \ldots, g_7, h_1, \ldots, h_7\rangle$, where g_i and h_i are, respectively, the fraction of XY_2 pairs of type i and the fraction of XY_2 triplets of type i. In Sec. 5, we demonstrate that such a motif-based attribute vector does a remarkable job of codifying and capturing the information content of the XY_2 polymeric class of systems, allowing us to train our machines and make high-fidelity predictions.

3.2. *Gene selection*

Gene selection is based on SVMs [14–18] (please, see Appendix B for more information). It takes as input n genes $\{g_1, g_2, g_3, \ldots, g_n\}$, and l vectors $\{v_1, v_2, v_3, \ldots, v_l\}$. As an example, each v_i could be an outcome of a microarray experiment and each vector could be of the following form: $v_i = \{x_i^1, x_i^2, x_i^3, \ldots, x_i^n, y_i\}$. Here x_i^j is the expression level of the j^{th} gene g_j in experiment i. The value of y_i is either $+1$ or -1 based on whether the event of interest is present in experiment i or not. The problem is to identify a set of genes $\{g_i^1, g_i^2, g_i^3, \ldots, g_i^m\}$ sufficient to predict the value of y_i in each experiment. Given a set of vectors, the gene selection algorithm learns to identify the minimum set of genes needed to predict the event of interest and the prediction function. These vectors form the training set for the algorithm. Once trained, the algorithm is provided with a new set of data which is called the test set. The accuracy of gene selection is measured in the test set as a percentage of microarray data on which the algorithm correctly predicts the event of interest. The procedure solely relies on the concept of SVM.

The gene selection algorithm of Song and Rajasekaran [27] is based on the ideas of combining the mutual information among the genes and incorporating correlation information to reject the redundant genes. The Greedy Correlation Incorporated Support Vector Machine (GCI-SVM) algorithm of [27] can be briefly summarized as follows: The SVM is trained only once and the genes are sorted according to the norm of the weight vector corresponding to these genes. Then the sorted list of genes are examined starting from the second gene. The correlation of each of these genes with the first gene is computed until one whose correlation with the first one

is less than a certain predefined threshold is found. At this stage this gene is moved to the second place. Now the genes starting from the third gene are examined and the correlation of each of these genes with the second gene is computed until a gene whose correlation with the second gene is less than the threshold is encountered. The above procedure is repeated until end of the list of the sorted genes is reached. In the last stage, genes based on this adjusted sorted genes are selected. GCI-SVM brings the concept of sort-SVM and RFE-SVM [9] altogether which makes it more efficient. In a nutshell, GCI-SVM works as follows:

- Compute the correlation coefficient for each pair of genes.
- Train the SVM using the training data set.
- Sort the genes based on their weight values.
- Go through the sorted genes, pick those genes whose correlation with the previously picked genes is less than a threshold.
- Move in order all picked genes to the front of the sequence; correspondingly, unpicked genes are moved to the end.

We incorporate *gene selection algorithm* in our RFS1 algorithm to effectively find the best subset of features from the entire set of features. We calculate the accuracy of the selected subset of features returned by RFS1 with the help of GSA. The search for the best subset of features proceeds based on the accuracy as stated in Algorithm 1 until a desired convergence is achieved. Please see Sec. 5 for more details.

3.3. *Data integration*

Data integration involves combining data residing in different sources and providing users with a unified view of these data [3]. As an example, the same person may have health care records with different providers. It helps to merge all the records with all the providers and cluster these records such that each cluster corresponds to one individual. Such an integration, for instance, could help us avoid performing the same tests again and hence save money. Several techniques [22–25] have been proposed to solve the data integration problem. Tian *et al.* [20] have proposed several space and time efficient techniques to integrate multiple datasets from disparate data sources. They employ agglomerative hierarchical clustering techniques [35] to integrate data of similar types and avoid the computation of cross-products. It can cope up with some common errors committed in input data such as typing distance and sound distance. Furthermore, it can deal with some human-made typing errors e.g., reversal of the first and last names, nickname usage, and attribute truncation.

We incorporate the data integration technique of [20] in our RFS1 algorithm. At each iteration RFS1 returns a subset of features such as name, sex, postal code, etc. from the different databases. Based on the selected features the data integration scheme of Tian *et al.* integrates data of similar types into clusters and calculates the accuracy of the clusters. RFS1 then proceeds with the search by exploiting the accuracy. Please, see Sec. 5 for details.

4. Our Algorithms

If we can identify a subset of the features that are the most important in determining a property, it will lead to computational efficiency as well as a better accuracy. It is conceivable that some of the features might be hurtful rather than helpful in predictions. Let $\vec{A} = |a_1, a_2, \ldots, a_n\rangle$ be the set of features under consideration. One could use the following simple strategy, in the context of any learning algorithm, to identify a subset of $\vec{A}$ that yields a better accuracy in predictions than $\vec{A}$ itself. For some small value of k (for example 2), we identify all the $\binom{n}{k}$ subsets of $\vec{A}$. For each such subset we train the learner, figure out the accuracy we can get, and pick that subset $\vec{S}$ that yields the best accuracy. Now, from the remaining features, we add one feature at a time to $\vec{S}$ and for each resultant subset, we compute the accuracy obtainable from the learner. Let $\vec{S}'$ be the set (of size $k+1$) of attributes that yields the best accuracy. Next, from the remaining attributes, we add one feature at a time to $\vec{S}'$ and identify a set of size $k+2$ with the best accuracy, and so on. Finally, from out of all of the above accuracies, we pick the best one.

4.1. *Randomized feature selector* 1 (*RFS*1)

We can think of the above simple technique as a greedy algorithm that tries to find an optimal subset of attributes and it may not always yield optimal results. On the other hand, it will be infeasible to try every subset of attributes (since there are 2^n such subsets). We propose the following novel approach instead: Consider the space of all possible subsets of attributes. We start with a random point p (i.e., a random subset of the features) in this space and calculate the accuracy q corresponding to this subset. We then flip an unbiased three sided coin with sides 1, 2, and 3. If the outcome of the coin flip is 1, we choose a random neighbor p' of this point by removing one feature from p and adding a new feature to p. After choosing p', we compute its accuracy q'. If $q' > q$ then we move to the point p' and proceed with the search from p'. On the other hand, if $q' < q$, then we stay with point p (with some probability u) or move to point p' with probability $(1-u)$. This step is done to ensure that we do not get stuck in a local maximum. If the outcome of the coin flip is 2, we choose a random neighbor p' by removing one feature from p and compute its accuracy q'. The next steps are the same as stated in the case of 1. Consider the last case where the outcome of the coin flip is 3. We choose a random neighbor p' by adding one feature to p and compute its accuracy q'. The rest of the steps are the same as above. If $q' > q$ then we move to the point p' and proceed with the search from p'. On the other hand, if $q' < q$, then we stay with point p (with some probability u) or move to point p' with probability $(1-u)$. We proceed with the search from the point we end up with. This process of searching the space is continued until no significant improvement in the accuracy can be obtained. A relevant choice for u is $\exp(-c(q-q'))$ for some constant c. In fact, the above algorithm resembles the simulated annealing (SA) algorithm of [13]. Note that our algorithm is very different from SA. In particular, our algorithm is much simpler than SA. Details of our algorithm can be found in Algorithm 1.

4.2. *Randomized feature selector* 2 (*RFS*2)

Another variation of the RFS1 algorithm is RFS2. It is based on pure random walk on the search space. In RFS1 if $q' < q$, then we stay with the point p (with some probability u) or move to point p' with probability $(1 - u)$ as described above. Instead in RFS2 if $q' < q$ we always stay with point p and restart the search from that point. Details of our algorithm can be found in Algorithm 2. Please, see line 19 which differentiates RFS2 and RFS1.

4.3. *Randomized feature selector* 3 (*RFS*3)

RFS3 is another variation of RFS1. In RFS3 we iterate the search from different points in the search space. In each iteration if $q' < q$ we randomly select a brand new subset from the search space and proceed with the search as described in RFS1. Details of our algorithm can be found in Algorithm 3. Please see lines 19–22 which differentiate RFS3 and RFS1.

5. Analysis of Our Algorithms

In this section we illustrate runtime analysis and convergence proof of RFS1 algorithm. The same analyses can be applied to RFS2 and RFS3 algorithms with some minor modifications. In the RFS1 algorithm it is easy to see that each run of the **repeat** loop takes $O(n)$ time (please, see the pseudocode of Algorithm 1). This can be reduced to $O(\log n)$ by keeping $F - F'$ as a balanced tree (such as a red-black tree). An important question is how many runs will be needed for convergence. In this section we answer this question. The analysis is based on representing the steps of the algorithm as a time homogeneous Markov chain. The analysis is similar to the one in [24].

We can conceive of a directed graph $G(V, E)$ for the feature selection problem as follows: Every subset of the n features is a node in G. From any node $p \in V$, there will be edges to its neighbors. Clearly, in the Algorithm RFS1, for every node p, there are three kinds of neighbors: Let p' be a neighbor of p. If p' is of the first kind, then the number of features in p and p' will be the same. Thus there are $N_p^1 \leq n$ such neighbors. If p' is of the second type, then p' will have one more feature than p. Finally, if p' is of the third type, then p' will have one less feature than p. As a result, it follows that there will be $\leq 3n$ neighbors for any node in the graph $G(V, E)$. Let the number of neighbors of p of the second and third types be N_p^2 and N_p^3, respectively.

Algorithm RFS1 starts from a random node p in G and performs a random walk in this graph. The next node visited can be of type 1, 2, or 3 all with equal probability. The next node visited depends only on the current node. We can thus model RFS1 as a time homogeneous Markov chain. In contrast, the simulated annealing algorithm has been modeled as a time inhomogeneous Markov chain (see e.g., [21] and [24]).

Algorithm 1: Randomized Feature Selector (RFS1)

Input: The set F of all possible features and an Inductive Learning Algorithm $\mathcal{L}$

Output: A near optimal subset F' of features

begin

1 Randomly sample a subset F' of features from F.

2 Run the inductive learning algorithm $\mathcal{L}$ using the features in F'.

3 Compute the accuracy A of the concept C learnt by $\mathcal{L}$. (Note that any learning algorithm leanrs a function (of the attributes). This function is what we refer to as the concept learnt).

4 **repeat**

5 Flip an unbiased three sided coin with sides 1, 2, and 3.

6 **if** (the outcome of the coin flip is 1){

7 Choose a random feature f from $F - F'$ and add it to F'.

8 Remove a random feature f' from F' to get F''.

9 } **else if** (the outcome of the coin flip is 2){

10 Choose a random feature f from $F - F'$ and add it to F' to get F''.

11 } **else if** (the outcome of the coin flip is 3){

12 Remove a random feature f from F' to get F''.

13 }

14 Run the inductive learning algorithm $\mathcal{L}$ using the features in F''.

15 Compute the accuracy A' of the concept C' learnt by $\mathcal{L}$.

16 **if** $(A' > A)$ {

17 $F' := F''$ and $A := A'$; Perform the search from F'.

18 } **else**{

19 With probability u perform the search from F' and with probability $1 - u$ perform the search from F'' with $A := A'$.

20 }

until *no significant improvement in the accuracy can be obtained*;

21 Output F'.

Note that for any two nodes p and p' in G, there is a directed path from p to p' and hence G is strongly connected. For any node p in G, let $q(p)$ be the accuracy

Algorithm 2: Randomized Feature Selector 2 (RFS2)

Input: The set F of all possible features and an Inductive Learning Algorithm $\mathcal{L}$
Output: A near optimal subset F' of features
begin
1 Randomly sample a subset F' of features from F.
2 Run the inductive learning algorithm $\mathcal{L}$ using the features in F'.
3 Compute the accuracy A of the concept C learnt by $\mathcal{L}$.
4 **repeat**
5 Flip an unbiased three sided coin with sides 1, 2, and 3.
6 **if** (the outcome of the coin flip is 1){
7 Choose a random feature f from $F - F'$ and add it to F'.
8 Remove a random feature f' from F' to get F''.
9 } **else if** (the outcome of the coin flip is 2){
10 Choose a random feature f from $F - F'$ and add it to F' to get F''.
11 } **else if** (the outcome of the coin flip is 3){
12 Remove a random feature f from F' to get F''.
13 }
14 Run the inductive learning algorithm $\mathcal{L}$ using the features in F''.
15 Compute the accuracy A' of the concept C' learnt by $\mathcal{L}$.
16 **if** $(A' > A)$ {
17 $F' := F''$ and $A := A'$; Perform the search from F'.
18 } **else**{
19 Perform the search from F'
20 }
until *no significant improvement in the accuracy can be obtained*;
21 Output F'.

of the feature set corresponding to p. If p is any node and p' is a neighbor of p of type k $(1 \leq k \leq 3)$, then the transition probability $P_{pp'}$ from p to p' is given by:

$$P_{pp'} = \begin{cases} 0 & \text{if } p' \notin N(p) \ \& \ p' \neq p \\ \frac{1}{3N_p^k} \min(1, \exp\{c(q(p') - q(p))\}) & \text{if } p' \text{ is a neighbor of } p \text{ of type } k \end{cases}$$

Algorithm 3: Randomized Feature Selector 3 (RFS 3)

Input: The set F of all possible features and an Inductive Learning Algorithm $\mathcal{L}$

Output: A near optimal subset F' of features

begin

Randomly sample a subset F' of features from F.

Run the inductive learning algorithm $\mathcal{L}$ using the features in F'.

Compute the accuracy A of the concept C learnt by $\mathcal{L}$.

repeat

  Flip an unbiased three sided coin with sides 1, 2, and 3.

  if (the outcome of the coin flip is 1){

    Choose a random feature f from $F - F'$ and add it to F'.

    Remove a random feature f' from F' to get F''.

  } **else if** (the outcome of the coin flip is 2){

    Choose a random feature f from $F - F'$ and add it to F' to get F''.

  } **else if** (the outcome of the coin flip is 3){

    Remove a random feature f from F' to get F''.

  }

  Run the inductive learning algorithm $\mathcal{L}$ using the features in F''.

  Compute the accuracy A' of the concept C' learnt by $\mathcal{L}$.

  if $(A' > A)$ {

    $F' := F''$ and $A := A'$; Perform the search from F'.

  } **else**{

    Randomly sample a subset F' of features from F.

    Run the learning algorithm $\mathcal{L}$ using the features in F'.

    Compute the accuracy A of the concept C learnt by $\mathcal{L}$.

    Perform the search from F'.

  }

until *no significant improvement in the accuracy can be obtained*;

Output F'.

and

$$P_{pp} = 1 - \sum_{p' \in N(p)} P_{pp'} .$$

RFS1 is said to have converged if the underlying Markov chain had been in a globally optimal state at least once. Let p be the starting node (i.e., start state) of the Markov chain and let p' be a globally optimal state. Then there is a path from p to p' of length $\leq n$.

Let $\Delta = \max_{p \in V,\ p' \in N(p)} \{q(p) - q(p')\}$. Also, let the degree and diameter of $G(V, E)$ be d and D, respectively. Clearly, $d \leq 3n$ and $D \leq n$. If p' is a neighbor of p, then $P_{pp'}$ is $\geq \frac{1}{3d} \exp(-c\Delta)$. We can derive a time bound within which the RFS1 algorithm will converge as stated in the following Lemma.

Lemma 1. *If p is any state in V, then the expected number of steps before a global optimal state is visited starting from p is $\leq \left(\frac{1}{3d} \exp(-c\Delta)\right)^{-D}$.*

Proof. Let g be any globally optimal state. Then there exists a directed path from p to g in $G(V, E)$ of length $\ell \leq D$. Let $e_1, e_2, \ldots, e_\ell$ be the sequence of edges in the path. The probability that g is visited starting from p is greater than or equal to the probability that each one of the edges $e_i,\ 1 \leq i \leq \ell$ is traversed in succession. The later probability is at least $\left[(\frac{1}{3d}) \exp(-c\Delta)\right]^{\ell} \geq \left[(\frac{1}{3d}) \exp(-c\Delta)\right]^{D}$. As a result, the expected number of steps before g is visited is $\leq [3d\ \exp(c\Delta)]^D$. □

Theorem 2. *RFS1 converges in time $\leq 2m[3d\ \exp(c\Delta)]^D$, with probability $\geq (1 - 2^{-m})$, independent of the start state (for any integer $m \geq 1$).*

Proof. Let $E = 2[3d\ \exp(c\Delta)]^D$. We prove by induction (on m) that the probability of a global optimal state g not being visited in mE steps is $\leq 2^{-m}$.

Induction Hypothesis. Independent of the start state, probability that g is not visited in mE steps is $\leq 2^{-m}$.

Base case (for $m = 1$) follows from Lemma 1 and Markov's inequality.

Induction step. Assume the hypothesis for all $m \leq (r - 1)$. We'll prove the hypothesis for $m = r$. Let $p_E, p_{2E}, \ldots, p_{(r-1)E}$ be the states of the Markov chain during time steps $E, 2E, \ldots, (r - 1)E$, respectively. Let A be the event: g is not visited during the first E steps, and B be the event: g is not visited during the next $(r - 1)E$ steps.

The probability P that g is not visited in rE steps is given by

$$P \ = \ Prob.[B/A] \ \times \ Prob.[A].$$

Since $Prob.[B/A]$ depends only on what state the Markov chain is in at time step E and the time duration $(r - 1)E$, we infer:

$$P = Prob.[A] \sum_{p \in V} Prob.[B/p_E = p] \times Prob.[p_E = p].$$

Applying the induction hypotheis, Prob.[A] is $\leq 1/2$ and Prob.[B/$p_E = p$] is $\leq 2^{-(r-1)}$ for each $p \in V$. Therefore, we have,

$$P \leq \frac{1}{2}\, 2^{-(r-1)} = 2^{-r}. \qquad \square$$

6. Results and Discussions

We have employed our randomized feature selection algorithms on three different application domains. These applications include but not limited to the prediction of properties of materials, processing of biological data, and data integration. Our algorithms are generic and can be used in conjunction with any learning algorithm.

6.1. *Materials property prediction*

We consider polymeric dielectrics created using the XY_2 blocks as described in Sec. 3. If we assume that our repeat unit consists of 4 building blocks, and that each building block can be any of 7 distinct units (namely, CH_2, SiF_2, $SiCl_2$, GeF_2, $GeCl_2$, SnF_2, and $SnCl_2$), we have a total of 175 distinct polymer chains (accounting for translational symmetry). Of these, we set 130 to be in the training set, and the remainder in the test set to allow for validation of the machine learning model.

Attribute vectors may be chosen in different ways. Consider the motif-based one as described in Sec. 3, i.e., our attribute vector, $\vec{A}^i = |f_1^i, \ldots, f_6^i, g_1^i, \ldots, g_7^i, h_1^i, \ldots, h_7^i\rangle$, where f_j^i, g_j^i and h_j^i are, respectively, the fraction of XY_2 units of type j, the fraction of pair clusters of XY_2 units of type j and the fraction of triplet clusters of XY_2 units of type j. Once our machine has learned how to map between the attribute vectors and the properties using the training set, we make predictions on the test set (as well as the training set). Furthermore, we considered several 8-block repeat units (in addition to the 175 4-block systems), and performed our machine learning scheme.

We have tested the above techniques on the KRR scheme presented in Appendix A with the systems represented using the motif-based attribute vectors. We refer to the greedy extension as the modified greedy KRR (mg-KRR) approach and the modified optimization version as mo-KRR. mg-KRR and mo-KRR are based on Sequential Forward Search (SFS) and Randomized Feature Selector (RFS1) algorithms, respectively. In each iteration of each of the algorithms (e.g. mg-KRR and mo-KRR) the accuracy of the selected subset is measured by employing the KRR scheme. Based on the accuracy it proceeds with the search until a desired accuracy/convergence achieved. An assessment of the improvement in the predictive power when mg-KRR and mo-KRR are used for the three properties of interest (namely, the band gap, the electronic part of the dielectric constant and the total dielectric constant) is presented in Table 1. As can be seen, the level of accuracy of the machine learning schemes is uniformly good for all three properties across the 4-block training and test set, as well as the 8-block test set, indicative of the high-fidelity nature of this approach. In particular, note that the mo-KRR method,

Table 1. KRR and modified KRR (mg-KRR and mo-KRR) schemes.

		Bandgap		Electric DC		Total DC	
System	Method	Accuracy	Features	Accuracy	Features	Accuracy	Features
4-Block	KRR	92.98%	20	93.75%	20	96.49%	20
	mg-KRR	93.07%	19	94.22%	11	97.23%	14
	mo-KRR	93.43%	16	94.23%	18	97.63%	14
8-Block	KRR	96.95%	20	90.58%	20	95.81%	20
	mg-KRR	96.95%	20	90.64%	15	95.99%	19
	mo-KRR	97.45%	17	95.17%	12	97.68%	13

Table 2. KRR, modified KRR, distributed KRR, and randomized KRR (mo-KRR, dis-KRR, and ran-KRR) schemes.

	Bandgap		Electric DC		Total DC	
Method	Accuracy	Features	Accuracy	Features	Accuracy	Features
KRR	92.85%	162	89.30%	162	87.01%	162
mo-KRR	94.07%	127	92.01%	125	88.57%	127
dis-KRR	93.56%	111	91.18%	118	87.63%	113
ran-KRR	96.48%	105	95.68%	102	94.47%	102

in general, leads to better accuracy. More importantly, typically, the number of attribute components decreases significantly. This means a significant reduction in the run times of the algorithm while predicting parameter values for an unknown material.

Furthermore to demonstrate the practical applicability we have tested all the versions of our algorithm under KRR scheme with the systems represented using the motif-based attribute vectors. Each vector contains 162 components of 6 units, 23 pairs, and 133 triplets. We refer to the modified optimization version as mo-KRR (based on RFS1 algorithm), modified randomized version as ran-KRR (based on RFS2 algorithm), and distributed version of mo-KRR as dis-KRR (based on RFS3 algorithm). In dis-KRR we iterate the search from different points in the search space. In each iteration we randomly select a brand new subset of components from 162 components and proceed with the search by executing mo-KRR. The number of iterations is selected in such a way that the run time of dis-KRR is the same as that of mo-KRR. When all the iterations are complete, dis-KRR outputs the best one from all the iterations. The results are shown in Table 2. In general all the versions of our algorithm lead to a better accuracy with a significant decrease in the number of attributes. Please note that real datasets are used to perform all the experiments to predict materials property.

6.2. *Gene selection*

We have used the gene selection algorithm to identify some of the best features that can together identify two groups. The gene selection algorithm has two phases. In the first phase, the algorithm is trained with a training dataset. In this phase the algorithm comes up with a model of concept. In the second phase of the algorithm

Table 3. GSA and modified GSA (GSA and mo-GSA) schemes.

		GSA		Modified GSA	
System	Method	Accuracy	Features	Accuracy	Features
Dataset 1	GAUSSIAN	50%	15	54%	10
	LINEAR	49%	15	62%	12
Dataset 2	GAUSSIAN	52%	20	60%	13
	LINEAR	53%	20	65%	13
Dataset 3	GAUSSIAN	49%	25	58%	9
	LINEAR	50%	25	58%	11
Dataset 4	GAUSSIAN	50%	30	59%	13
	LINEAR	56%	30	62%	13

a test dataset is presented. The model learned in the first phase is used to classify the elements residing in the test dataset. As a result, the accuracy of the model learned can be computed. At first, we generated 4 simulated datasets each having 200 subjects with 15, 20, 25, and 30 features, respectively. Each of the features has been given a random value in the range $[0, 99]$. We then randomly assigned a class label to each of the subjects residing in each dataset. Specifically, each subject is assigned to group 1 with probability $\frac{1}{2}$ and it is assigned to group 2 with probability $\frac{1}{2}$. We trained the classifier using a training set which consists of 50 percent of data from each of group 1 and group 2 (data being chosen randomly). The test set is formed using the other 50 percent from group 1 and group 2, respectively.

At first GSA is trained with the training set and it builds a model for the concept being learnt using SVMs [27]. We have used LINEAR, and GAUSSIAN RBF kernel functions in SVM to build the model. Using the test data we have measured the accuracy by employing the model built. We then employ our randomized feature selection algorithm RFS1 on each of the datasets. We call this scheme as modified GSA (mo-GSA). In each iteration mo-GSA selects a subset of features as described in Algorithm 1. The accuracy of the selected subset is measured by GSA. Based on the accuracy, mo-GSA proceeds with the search until a desired accuracy/convergence achieved. From the results shown in Table 3 it is evident that after employing mo-GSA the accuracy has greatly improved and at the same time the number of features has decreased significantly with respect to GSA.

6.3. *Data integration*

Data integration technique of Tian *et al.* [20] is used to detect similar types of data from a set of databases. To test the performance of our approach, we generated 4 artificial datasets each having 10,000 subjects where each subject has 5 features. The features consist of a person's first name, last name, date of birth, sex, and zip code. In general, each person has multiple records. Since errors are introduced randomly in the features, instances of the same individual may differ from each other. Accuracy of any data integration method is calculated as the fraction of persons for whom all the instances have been correctly identified to be belonging to the same person.

Table 4. DI and modified DI (DI and mo-DI) schemes.

	Data Integrator		Modified Data Integrator	
System	Accuracy	# of Features	Accuracy	# of Features
Dataset 1	46.72%	5	89.71%	2
Dataset 2	85.50%	5	90.31%	3
Dataset 3	85.51%	5	90.32%	4
Dataset 4	85.50%	5	86.61%	3

At first we execute the algorithm proposed by [20] directly on each of the datasets and calculate the accuracy as defined above. We call this scheme as Data Integrator (DI). We then employ our randomized feature selection algorithm RFS1 on each of the datasets. We call this scheme as modified Data Integrator (mo-DI). In each iteration mo-DI randomly selects a subset of features as described in Algorithm 1. The accuracy of the selected subset is measured by [20]. Based on the accuracy mo-DI proceeds with the search until a desired accuracy/convergence is achieved. From the results shown in Table 4 it is evident that after employing mo-DI the accuracy has greatly improved and at the same time the number of features has also decreased with respect to DI.

6.4. *The comparisons*

We compared RFS1 algorithm with both of wrapper and filter-based schemes. At first, we generated a simulated dataset having 200 subjects with 15 features. Each of the features was given a random value in the range $[0, 99]$. We then randomly assigned a class label to each of the subjects residing in the dataset. Specifically, each subject was assigned to class 1 with probability $\frac{1}{2}$ and it was assigned to class 2 with probability $\frac{1}{2}$. We further divided the dataset into two parts namely control and test sets. Control set consists of 50 percent of data from each of group 1 and group 2 (data being chosen randomly). The test set was formed using the other 50 percent from group 1 and group 2, respectively.

The comparison procedure had two phases. In the first phase, the feature selection algorithm of interest selected the best feature subset from the control set. In the second phase the accuracy was calculated with the help of GSA from the test set considering only the subset of features given by the first phase. In the case of filter-based methods, we took the best subset of features in such a way that the size of the subset was identical to the size of the subset returned by mo-GSA. On the other hand wrapper-based methods gave the best subset from the entire space of features. In our comparisons, the filter-based methods we had employed Chi-square, correlation-based (e.g., linear and rank), and entropy-based (e.g., information and gain-ratio) filters. The wrapper methods used is CFS [38]. This algorithm makes use of best first search for searching the attribute subset space. Information on the implementation details of these algorithms can be found in [6]. The comparison results show that our feature selection algorithm (specifically, mo-GSA) outperforms some of the best known algorithms existing in the current literature. Please see

Table 5. The Comparisons.

Method Name	Class	% Accuracy	# of Features
mo-GSA	*wrapper*	54%	10
Chi-squared Filter	*filter*	48%	10
Linear Correlation-based Filter	*filter*	51%	10
Rank Correlation-based Filter	*filter*	51%	10
Information Gain-based Filter	*filter*	46%	10
Gain Ratio-based Filter	*filter*	46%	10
CFS Filter	*wrapper*	50%	2

Table 5 for details. As the datasets and the class labels were randomly generated from a uniform distribution. The correlations among the subjects in the same class are very low. The accuracies of the selected features given by the feature selection algorithms of interest reflect this fact. As the correlations are very low, the prediction model built by the learning algorithms will also not be sufficiently accurate to classify the subjects from unseen data.

7. Conclusions

We have presented three novel randomized search techniques which are generic in nature and can be applied to any inductive learning algorithm for selecting a subset of the most relevant features from the set of all possible features. The proposed schemes fall into the class of wrapper methods where the prediction accuracy in each step is determined by the learning algorithm of interest. To demonstrate the validity of our approaches, we have applied it in three different applications, namely, biological data processing, data integration, and materials property prediction. It is evident from the simulation results shown above that our proposed techniques are indeed reliable, scalable, and efficient. Our experiments also reveal that our feature selection algorithms perform better than some of the best known algorithms existing in the current literature.

Acknowledgment

This work has been supported in part by the following grant: NIH R01-LM010101.

References

[1] P. Christen, and K. Goiser K, *Quality and complexity measures for data linkage and deduplication*, In Quality Measures in Data Mining. Volume 43. Edited by Guillet F, Hamilton H. New York: Springer, 2007, pp. 127151.

[2] C. Cortes, and V. Vapnik, *Support Vector Networks*, In Machine Learning, 20: 1-25, 1995.

[3] *Data integration*, In http://en.wikipedia.org/wiki/Data_integration.

[4] J. Doak, *An Evaluation of Feature Selection Methods and Their Application to Computer Security*, In Technical report, Univ. of California at Davis, Dept. Computer Science, 1992.

[5] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios, *Duplicate Record Detection: A Survey*, In IEEE Trans Knowl Data Eng, 19:116, 2007.

[6] FSelector, *Optimization by simulated annealing*, In http://cran.r-project.org/web/packages/FSelector/FSelector.pdf.

[7] S. Geman, E. Bienenstock, and R. Doursat, *Neural networks and the bias/variance dilemma*, In Neural Computation 4(1), pp. 1-58, 1992.

[8] C.-W. Hsu, and C.-J. Lin, *A Comparison of Methods for Multiclass Support Vector Machines"*, In IEEE Transactions on Neural Networks, 2002.

[9] G. Isabelle, J. Weston, S. Barnhill, and V.N. Vapnik, *Gene Selection for Cancer Classification using Support Vector Machines*, In Machine Learning, 46, pp. 389-422, 2002.

[10] A. Jain, and D. Zongker, *Feature selection: evaluation, application, and small sample performance*, In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pp. 153-158, 1997.

[11] T. Jirapech-Umpai, and S. Aitken, *Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes*, In BMC Bioinformatics, 6:148, 2005.

[12] T. Joachims, *Transductive Inference for Text Classification using Support Vector Machines*, In 1999 International Conference on Machine Learning (ICML), pp. 200-209, 1999.

[13] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, *Optimization by simulated annealing*, In *Science* 220(4598), pp. 671-680, 1983.

[14] J. Kittler, *Feature set search algorithm, in C.H.Chen, Ed., Pattern Recognition and Signal Processing*, In Sijthoff and Noordhoff, Alphen aan den Rijn, Netherlands, pp.41-60, 1978.

[15] R. Kohavi, and G.H. John, *Wrappers for Feature Subset Selection*, In Artificial Intelligence, vol. 97, nos. 1-2, pp. 273-324, 1997.

[16] M. Kudo, and J. Sklansky, *Comparison of algorithms that select features for pattern classifiers*, In Pattern Recognition 33, pp. 25-41, 2000.

[17] Y. Lee, Y. Lin, and G. Wahba, *Multicategory Support Vector Machines, Theory, and Application to the Classification of Microarray Data and Satellite Radiance Data*, In J. Amer. Statist. Assoc. 99, Issue 465: 67-81, 2004.

[18] H. Liu, and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining* In Boston: Kluwer Academic, 1998.

[19] C.S. Liu, G. Pilania, C. Wang, and R. Ramprasad, *How critical are the van der Waals interactions in polymer crystals?*, In *J. Phys. Chem.* C, 116, 9347, 2012.

[20] T. Mi, S. Rajasekaran, and R. Aseltine, *Efficient algorithms for fast integration on large data sets from multiple sources*, In BMC Med Inform Decis Mak, 12:59, 2012.

[21] D. Mitra, F. Romeo, and A.S. Vincentelli, *Convergence and Finite-Time Behavior of Simulated Annealing*, In Advances in Applied Probability, Sept. 1986.

[22] P.M. Narendra, and K.A. Fukunaga, *Branch and Bound Algorithm for Feature Subset Selection*, In IEEE Trans. Computer, vol. 26, no. 9, pp. 917-922, Sept. 1977.

[23] P. Pudil, J. Novovicova, and J. Kittler, *Floating search methods in feature selection*, In Pattern Recognition Letters, vol. 15, pp. 1119-1125, 1994.

[24] S. Rajasekaran, *On Simulated Annealing and Nested Annealing*, In Journal of Global Optimization, 16(1), 2000, pp. 43–56.

[25] J.S. Russell, and N. Peter, *Artificial Intelligence: A Modern Approach* (2*nd ed.*), In Prentice Hall, Upper Saddle River, NJ, pp. 111-114, ISBN 0-13-790395-2, 2003.

[26] C. Saunders, A. Gammerman, and V. Vovk, *Ridge Regression Learning Algorithm in Dual Variables*, In 15th International Conference on Machine Learning, Madison, WI, pp. 515-521, 1998.

[27] M. Song, and S. Rajasekaran, *A greedy correlation-incorporated SVM-based algorithm for gene selection*, In Proc. of AINA Workshops, pp. 657-661, 2007.
[28] Y. Sun, S. A. Boggs, and R. Ramprasad, *The intrinsic electrical breakdown strength of insulators from first principles*, In *Appl. Phys. Lett* 101, 132906, 2012.
[29] A.A. Tikhonov, and V.Y. Arsenin, *Solutions of ill-posed problems*, In New York: John Wiley, 1977.
[30] V.N. Vapnik, *The Nature of Statistical Learning Theory*, In Springer, 1995.
[31] C.C. Wang, G. Pilania, and R. Ramprasad, *Dielectric properties of carbon, silicon and germanium based polymers: A first principles study*, In *Phys. Rev.* B, under review.
[32] W.E. Winkler, *Overview of Record Linkage and Current Research Directions*, In [http://www.census.gov/srd/papers/pdf/rrs2006-02.pdf].
[33] W.E. Winkler, *Improved Decision Rules In The Fellegi-Sunter Model Of Record Linkage*, In Survey Research Methods, American Statistical Association. Volume 1, Alexandria, VA: American Statistical Association, pp. 274279, 1993.
[34] S. Saha, S. Rajasekaran *et al. Efficient techniques for genotypephenotype correlational analysis*, In BMC medical informatics and decision making, 13:41, 2013.
[35] S. Rajasekaran and S. Saha, *A Novel Deterministic Sampling Technique to Speedup Clustering Algorithms*, In Advanced Data Mining and Applications (ADMA), Hangzhou, China, pp. 34-46, 2013.
[36] S. Salzberg *et al.*, *Microbial gene identification using interpolated markov models*, Nucleic Acids Res., 26, 544548, 1998.
[37] M. Ben-Bassat, *Pattern recognition and reduction of dimensionality*, In P. Krishnaiah and L. Kanal, (eds.) Handbook of Statistics II, Vol. 1. North-Holland, Amsterdam. pp. 773-791, 1982.
[38] M. Hall, *Correlation-based feature selection for machine learning*, PhD Thesis, Department of Computer Science, Waikato University, New Zealand, 1999.

Appendix A. Kernel Ridge Regression (KRR)

Kernel ridge regression is a data-rich non-linear forecasting technique. It is applicable in many different contexts ranging from optical character recognition to business forecasting. KRR has proven to be better than many well-known predictors. It is not much different from ridge regression rather it employs a clever algebraic trick to improve the computational efficiency. The central idea in kernel ridge regression is to employ a flexible set of nonlinear prediction functions and to prevent over-fitting by penalization. It is done in such a way that the computational complexity is reduced significantly. This is achieved by mapping the set of predictors into a high-dimensional (or even infinite-dimensional) space of nonlinear functions of the predictors. A linear forecast equation is then estimated in this high dimensional space. It also employs a penalty (or shrinkage, or ridge) term to avoid over-fitting. It is called kernel ridge regression since it uses the kernel trick to map the set of predictors into a high dimensional space and adds a ridge term to avoid over-fitting.

Assume that we are given N observations (x_1, y_1), $(x_2, y_2), \ldots, (x_N, y_N)$ with $x_i \in \Re^d$ and $y_i \in \Re$, for $1 \leq i \leq N$. Our goal is to find a function f such that $f(x_i)$ is a good approximation of y_i for $1 \leq i \leq N$. Once we identify such a function we can use it on any unknown observation $x' \in \Re^d$ to estimate the corresponding y' as $f(x')$. Ridge regression calculates the parameter vector $w \in \Re^d$ of a linear model

$f(x) = w.x$ by minimizing the objective function:

$$W_{RR}(w) = \frac{1}{2} \parallel w\parallel^2 + \frac{\gamma}{N}\sum_{i=1}^{N}(y_i - w_i.x_i)^2. \tag{A.1}$$

The objective function used in ridge regression (1) implements a form of Tikhonov regularisation [29] of a sum-of-squares error metric, where γ is a regularization parameter controlling the bias-variance trade-off [7].

A non-linear form of ridge regression [26] can be obtained by employing the kernel trick. Here a linear ridge regression model is constructed in a higher dimensional feature space induced by a non-linear kernel function defining the inner product:

$$K(x_a, x_b) = \varphi(x_a).\varphi(x_b). \tag{A.2}$$

The kernel function can be any positive definite kernel. One of the popular kernels is Gaussian radial basis function (RBF) kernel:

$$K(x_a, x_b) = \exp\left(-\frac{\parallel x_a - x_b\parallel^2}{2\sigma^2}\right) \tag{A.3}$$

where σ is a tunable parameter. The objective function minimized in kernel ridge regression can be written as:

$$W_{KRR}(w) = \frac{1}{2} \parallel w\parallel^2 + \frac{\gamma}{N}\sum_{i=1}^{N}\xi_i^2 \tag{A.4}$$

subject to the constraints:

$$\xi_i = y_i - w.\varphi(x_i), \forall i \in \{1, 2, \ldots, N\}.$$

The output of the KRR model is given by the equation:

$$f(x) = \sum_{i=1}^{N}\alpha_i\varphi(x_i).\varphi(x) = \sum_{i=1}^{N}\alpha_i K(x_i, x). \tag{A.5}$$

Appendix B. Support Vector Machine (SVM)

Support Vector Machine has been developed by Vapnik *et al.* at AT&T Bell Laboratories [2, 30] which is the basis of any gene selection algorithm. Kernel-based techniques (such as support vector machines, Bayes point machines, kernel principal component analysis, and Gaussian processes) represent a major development in machine learning algorithms. Support vector machines (SVMs) are a group of supervised learning methods that can be applied to classification or regression. They represent an extension to nonlinear models of the generalized portrait algorithm. The basic idea is to find a hyperplane which separates any given d-dimensional data perfectly into two classes. Assume that we are given l training examples $\{x_i, y_i\}$, where each example has d inputs ($x_i \epsilon \Re^d$), and a class label $y_i \epsilon \{-1, 1\}$ ($1 \leq i \leq l$).

Now, all the hyperplanes in $\Re^d$ are parameterized by a vector (w), and a constant (b), expressed in the equation:

$$w \cdot x + b = 0\,. \tag{B.1}$$

Here x is a point on the hyperplane, w is a n-dimensional vector perpendicular to the hyperplane, and b is the distance of the closest point on the hyperplane to the origin. Any such hyperplane (w, b) that separates the data leads to the function:

$$f(x) = sign(w \cdot x + b)\,. \tag{B.2}$$

The hyperplane is found by solving the following problem:

$$\text{Minimize } J = \tfrac{1}{2} \parallel w\|^2; \text{ subject to } y_i(w \cdot x_i + b) - 1 \geq 0, \text{ where } i = 1, \ldots, l.$$

To handle datasets that are not linearly separable, the notion of a "kernel induced feature space" has been introduced in the context of SVMs. The idea is to cast the data into a higher dimensional space where the data is separable. To do this, a mapping function $z = \phi(x)$ is defined that transforms the d dimensional input vector x into a (usually higher) d' dimensional vector z. Whether the new training data $\{\phi(x_i), y_i\}$ is separable by a hyperplane depends on the choice of the mapping/kernel function. Some useful kernel functions are "polynomial kernel" and "GAUSSIAN RBF kernel". The *polynomial kernel* takes the form:

$$K(x_a, x_b) = (x_a \cdot x_b + 1)^p \tag{B.3}$$

where p is a tunable parameter, which in practice varies from 1 to ~ 10. Another popular one is the Gaussian RBF Kernel:

$$K(x_a, x_b) = \exp\left(-\frac{\parallel x_a - x_b\|^2}{2\sigma^2}\right) \tag{B.4}$$

where σ is a tunable parameter. Using this kernel results in the classifier:

$$f(x) = sign\left[\sum_i \alpha_i y_i \exp\left(-\frac{\parallel x - x_i\|^2}{2\sigma^2}\right) + b\right] \tag{B.5}$$

which is a Radial Basis Function, with the support vectors as the centers. More details and applications of SVM can be found in [8, 12, 17].