

## Chapter 2

# Feature Selection

**Abstract** Feature selection (FS) is important in machine learning tasks because it can significantly improve the performance by eliminating redundant and irrelevant features while at the same time speeding up the learning task. Given  $N$  features, the FS problem is to find the optimal subset among  $2^N$  possible choices. This problem quickly becomes intractable as  $N$  increases. In the literature, suboptimal approaches based on sequential and random searches using evolutionary methods have been proposed and shown to work reasonably well in practice. This chapter describes the mainstream feature selection technique theories.

**Keywords** Redundant and irrelevant features · Feature interaction · Feature quality evaluation · Feature selection models · Over-fitting learning models

### 2.1 Introduction

FS is the process of choosing a subset of features that improves the ML method performance. Formally, for  $N$  data samples with  $M$  features in each data sample, FS problem is to find from the  $M$ -dimensional observation space  $\mathbb{R}^M$ , a subspace of  $m$  features  $\mathbb{R}^m$  that best predict the target. This is achieved by reducing the number of features to remove irrelevant, redundant, or noisy information from the feature set.

FS primarily aids in alleviating the curse of dimensionality and speeding up the learning task. In practice, it also helps in optimizing data collection methodologies by identifying which data to collect. Therefore, FS is a key pre-processing step (see Fig. 1.3) to improved predictions.

In a typical ML problem, there is usually an optimal number of features that provide the minimum error (highest accuracy). Because the total number of subspaces is  $2^M$ , finding an optimal feature subset is usually intractable [1] and many problems related to FS have been shown to be NP-hard [2]. Alternatively, many sequential and random searches have been proposed.

### ***2.1.1 Redundant and Irrelevant Features***

An irrelevant feature carries no useful information in describing the relationships of the underlying data. However, a feature that is irrelevant by itself may become useful when considered in combination with some other features, e.g. learning the 2-input XOR function given one input is not possible hence one input is irrelevant by itself but the 2 inputs in combination can produce the decision function. The elimination of redundant features is another aspect in FS. This can only be addressed by considering feature subsets and not by individual feature relevance assessment. If highly correlated features are present, individual features may exhibit similar performance to the collective feature subset hence using only the non-redundant features will improve performance. Perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them, and high variable correlation (or anti-correlation) does not imply the absence of variable complementarity [3].

Redundant features can be handled by dimensionality reduction techniques such as principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA). Linear techniques such as PCA perform a linear mapping of the data to a lower dimensional space in such a way, the variance of the data in the low-dimensional representation is maximized. KPCA and manifold techniques are able to perform non-linear dimensionality reduction. A good review can be found in Ref. [4].

### ***2.1.2 Feature Interaction***

In the presence of multiple interacting features, the correlation of individual features with the target variable may not be significant. However, if such interacting features are considered in combination, they could show a strong relationship to the target variable. Jakulin and Bratko [5] define an interacting subset as an irreducible whole, i.e. removing a variable impedes prediction ability. While Jakulin and Bratko introduce feature selection algorithms to deal with 2-way (one feature and the class) and 3-way (two features and the class) interactions, other algorithms such as INTERACT can detect n-way interactions in a subset. It is questionable if real interacting feature subsets can be efficiently discovered but in our experience, the wrapper techniques discussed later lead to good results at the expense of computational time.

### ***2.1.3 Over-Fitting and the Optimal Number of Features***

Given a finite number of training samples with redundant features, the error rate of a typical learner will drop and then increase as the number of features is increased. Hence, in a typical problem, there exists an optimal number of features that provide

the minimum error (highest accuracy). It is easy to over-fit a model by including too many degrees of freedom which will lead to poor generalization. The importance of having a separate training set for FS has been investigated for regression and classification [6]. The authors of [6] point out if the same dataset is used for feature selection and the learning task, the relationship between the features and the target variables may appear much stronger than actuality. This is called “feature selection bias”. Cross validation is therefore a key aspect in effective FS and validation sets are important for model optimization such as model parameter-tuning and kernel selection (in kernel methods). K-fold cross validation, two-fold cross validation, leave-one-out cross validation and repeated random sampling cross validation are some popular cross validation techniques [7]. Time-series cross validation is slightly different because the data are not independent and, due to the correlations with other observations, leaving out an observation does not remove all the associated information. Cross validation can be used to choose the number of features that provides the best accuracy in the training (in-sample) data (see Chap. 5). Unfortunately, this does not ensure that the same features are good for testing (out-sample) data.

#### ***2.1.4 Feature Quality Evaluation***

The performance of a ML task before and after feature selection can be evaluated to verify that the FS algorithm has served its purpose. A FS algorithm should not only strive to improve the prediction performance, but also consider other important attributes such as the complexity of the resulting model, interpretability of the model, subset cardinality, computational time for feature selection, scalability and generalization of the resulting model.

## **2.2 Feature Selection Models**

Three feature selection models are considered: filter, wrapper and embedded. Filter models are independent of the learning algorithm and hence solely data dependent. Wrappers optimize with the specific learning function in the loop, assessing the performance of different feature subsets based on a cost function. Embedded models are built into the learning algorithm itself, for example, via a specific way of determining the weights in a neural network or a modified penalty term in a kernel based model function. Selected techniques from each category are reviewed in this brief. Feature ranking assigns a weight for each feature and feature subset selection evaluates different feature combinations. In general, filter models are used as feature ranking strategies and wrapper and embedded models are used as feature subset selection strategies.

### 2.2.1 Filter Models

A filter model makes use of intrinsic characteristics in the data for feature ranking. The feature ranking criterion can be based on dependency measures, distance measures and consistency measures. In this research, the commonly used filters include the information gain (dependency), maximum-relevance-minimal-redundancy (dependency), Pearson's correlation (dependency) and the Relief algorithm (distance). By evaluating these measures between the feature and the target variable the features can be assigned a weight and they can be ranked.

1. **Correlation:** Also called similarity measures or dependency measures. In time-series prediction tasks on stationary data, tests such as a Granger causality test can be used as a tool to aid in deciding whether one time-series can be useful in predicting another. If  $X$  and  $Y$  are two time-series,  $X$  is said to Granger-cause  $Y$  if  $Y$  can be better predicted using the histories of both  $X$  and  $Y$  than it can by using the history of  $Y$  alone. This is, in fact, a form of feature selection.

A good individual feature is highly correlated with the target so correlation measures can be used for ranking or subset selection. As an example, for a candidate feature  $x_i \in X$  and regression target  $Y$ , the Pearson correlation coefficient is given by  $\rho(x_i, Y) = \frac{\text{cov}(x_i, Y)}{\sqrt{\sigma(x_i)\sigma(Y)}}$  where  $\text{cov}$  is the covariance, and  $\sigma$  the variance.

Direct feature-target correlation by itself is however considered as a poor technique because a high feature-target correlation can still be an irrelevant feature. These are known as spurious correlations.

2. **Information Gain:** Information theoretic criteria can be used in place of correlation. Information (the negative of entropy) contained in a discrete distribution of feature  $X$  is given by,

$$H(X) = - \sum_i p(x_i) \log_2 p(x_i) \quad (2.1)$$

where the  $x_i$ s are the discrete feature values and  $p(x_i)$  is its probability. Continuous features are either discretized, or integration instead of summation is performed by fitting a kernel function to approximate the density of the feature  $X$ . Information embedded in the joint distribution is provided by,

$$H(Y, X) = - \sum_i \sum_j p(y_j, x_i) \log_2 p(y_j, x_i) \quad (2.2)$$

where  $p(y_j, x_i)$  is the joint probability. Mutual information (MI) provides a good measure of feature importance. MI which is calculated as  $MI(Y, X) = H(Y) + H(X) - H(Y, X)$  is equal to the Kullback-Leibler divergence given by,

$$MI(Y, X) = - \sum_{i,j} p(y_j, x_i) \log_2 \frac{p(y_j, x_i)}{p(y_j)p(x_i)} \quad (2.3)$$

A feature is more important if the mutual information  $MI(Y, X)$  between the target and the feature distributions is larger. Information gain is a similar criterion where  $IG(Y, X) = H(Y) - H(Y|X)$  is the difference between information contained in the class distribution  $H(Y)$ , and information after the distribution of feature values is taken into account, i.e. the conditional information  $H(Y|X)$ .

3. **Maximum-relevance-minimal-redundancy (mRMR):** mRMR is a feature selection technique that attempts to constrain features to a subset which are mutually as dissimilar to each other as possible, but as similar to the classification variable as possible. Maximum relevance features are assessed by the following equation where  $S$  is a feature set of  $m$  features ( $|S| = m$ );  $x_i$  is an individual feature;  $c$  is the target class and  $IG(x_i, c)$  is the difference between information contained in the target distribution  $H(c)$ , and conditional information  $H(c|x_i)$  as discussed above:

$$\max_{S, c} D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i, c). \quad (2.4)$$

Although such features are highly relevant, they could also be highly redundant. A criterion to select mutually exclusive features can be written as:

$$\min_S R = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i, x_j) \quad (2.5)$$

The “maximum-relevance-minimal-redundancy” criterion combines the above two goals by optimizing the cost function:

$$\max_{D, R} \Phi = D - R \quad (2.6)$$

Starting from an initial feature set  $S_{m-1}$  with  $m - 1$  features, the objective is to select the  $m$ th feature from the remaining set  $\{X - S_{m-1}\}$  by maximizing  $\Phi(\cdot)$ . This is done incrementally via Eq. 2.7 to find the solution.

$$\max_{x_j \in X - S_{m-1}} \left[ I(x_j, c) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j, x_i) \right] \quad (2.7)$$

4. **Relief Score:** Distance based filter models such as Relief ranks features that distinguish classes based on how well a feature can separate classes. The original Relief algorithm, proposed by Kira and Rendell [8], is a two-class filtering algorithm for features normalized to  $[0, 1]$ . Each feature is initially assigned a zero weight. An  $A$ -dimensional training example  $R$  is chosen randomly and the Euclidean distance to all other instances calculated. Denote the nearest hit in the same class  $H$ , and the nearest miss in a different-class  $M$ . Since a good feature

$R[A]$  should be able to separate class values, it should have a small distance to  $H$  and a large distance to  $M$ . Hence  $W[A]$  is adjusted to reward good features and penalize poor ones. The final selection of features is made by selecting those large  $W[A]$ , i.e. those that exceed a given threshold. The pseudo code for the Relief algorithm is given below:

---

**Algorithm 1** Relief Algorithm
 

---

```

1: set all weights  $W$  to 0;
2: for  $i := 1:m$  do ▷  $m$  is the training set size
3:   randomly select an instance  $R$ ;
4:   find nearest hit  $H$  and nearest miss  $M$ ;
5:   for  $A := 1:a$  do ▷  $a$  is the dimensionality of the feature vector
6:      $W[A] := W[A] - \text{diff}(R[A], H[A]) + \text{diff}(R[A], M[A])$ ;
7:   end for
8: end for
  
```

---

Different diff functions can be used for discrete e.g.  $\text{diff}(x, y) = 0$  if  $x$  and  $y$  are in the same class, 1 otherwise, and continuous feature values, e.g.  $\text{diff}(x, y) = (x - y)^2$ . RReliefF (regression ReliefF) was developed later for regression problems. Since regression applies to continuous variables, RReliefF uses a probability measure which is modeled by target variable values of instances belonging to different classes. Two disadvantages associated with Relief algorithms are (i) they are computationally expensive and (ii) they may fail to remove redundant features.

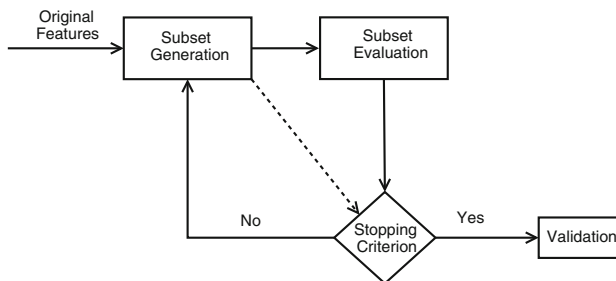
5. **Consistency-based filters:** Consistency measures attempt to find a minimum number of features that distinguish between the classes as consistently as the full set of features. An inconsistency arises when multiple training samples have the same feature values, but different class labels. Dash [9] presents an inconsistency-based FS technique called Set Cover. An inconsistency count is calculated by the difference between the number of all matching patterns (except the class label) and the largest number of patterns of different class labels of a chosen subset. If there are  $n$  matching patterns in the training sample space and there are  $c_1$  patterns belonging to class 1,  $c_2$  patterns belonging to class 2 and  $c_3$  patterns belonging to class 3, and if the largest number is  $c_2$ , the inconsistency count would be defined as  $n - c_2$ . By summing all the inconsistency counts and averaging over the size of the training sample size, a measure called the inconsistency rate for a given subset is defined. This is advantageous because (i) a feature subset can be evaluated in linear time (ii) can eliminate both irrelevant and redundant features (iii) some noise is reflected as a percentage of inconsistencies and (iv) unlike measures like correlation, information theoretic criterion and distance, this is a multivariate measure. Despite these advantages, it can be shown that Set Cover has a tendency to choose features that are highly correlated with the target, although in reality they can be irrelevant. For large dimensional feature spaces, filtering can offer reduced computational time compared to wrappers but this is dependent

on the sample size and the number of features. Other methods can also be applied following filtering. However, the question remains on which relevance measures should be used in a particular application.

In the presence of multiple interacting features, the individual feature relationship with the target can be insignificant but if such interacting features are considered in combination, a strong relationship to the target variable *may* be identified. In other words, “the  $m$  best features are not the best  $m$  features” [10]. Many efficient FS algorithms assume feature independence, ignoring the interacting features. Reference [5] defines an interacting feature subset as an irreducible whole: “a whole is reducible if we can predict it without observing all the involved variables at the same time”. Therefore, different feature combinations are often evaluated using wrapper models. For large dimensional feature spaces, a filter model is usually applied first to reduce the feature space dimensionality followed by the application of a wrapper model.

### 2.2.2 Wrapper Models

A wrapper model involves a feature evaluation criterion, a search strategy and a stopping criterion as depicted in Fig. 2.1. Unlike the filter model, a wrapper model utilizes the performance of the learning algorithm as the evaluation criterion. This ensures that the selected feature subset is well matched to the ML algorithm. Wrappers are computationally expensive since they require the ML algorithm to be executed in every iteration, and are usually used for low dimensional datasets. Search strategies generate different feature combinations to traverse through the feature space (generating feature subsets for evaluation). A widely used stopping criterion is to stop generating and evaluating new feature subsets when adding or removing features does not make any performance improvements [2]. The sequential steps of a wrapper model with  $X$  being the training data,  $A$  the learning algorithm, and  $S_0$  the initial subset of features as given in Ref. [2] is provided in Algorithm 2.



**Fig. 2.1** A general wrapper model [2]

---

**Algorithm 2** Wrapper Approach
 

---

**Input**

$X$  ▷ Training data with  $N$  features,  $|X| = n$   
 $N$  ▷ Number of features (feature space dimensionality)  
 $n$  ▷ Number of training examples  
 $A$  ▷ Learning Algorithm  
 $SC$  ▷ Stopping criterion  
 $S_0$  ▷ Initial subset of features

**Output**  $S_{best}$ 

▷ Selected feature subset

$S_{best} \leftarrow S_0$  ▷ Assign the current best feature subset as the initial subset  
 $\gamma_{best} \leftarrow eval(S_0, X, A)$   
**repeat**  
    $S \leftarrow SearchStrategy(X)$  ▷ Generate next subset to be evaluated  
    $\gamma \leftarrow eval(S, X, A)$   
   **if**  $\gamma \geq \gamma_{best}$  **then**  
      $\gamma_{best} \leftarrow \gamma$   
      $S_{best} \leftarrow S$   
   **end if**  
**until**  $SC$  is reached **return**  $S_{best}$

---

Search strategies generate different feature combinations to traverse through the feature space (generating feature subsets for evaluation).

For  $N$  features there exist  $2^N$  potential subsets. For even modest values of  $N$ , an exhaustive search over this huge space is intractable. The feature search therefore plays an important role in wrappers. Exponential, sequential and randomized searches are described next.

### 2.2.2.1 Feature Search

1. **Exponential search:** An exponential search returns the optimal subset. Although the order of the search space is  $\mathcal{O}(2^N)$ , the search need not be exhaustive, i.e. heuristics can be introduced to reduce the search space without compromising the chances of discovering the optimal subset [2], e.g. branch and bound and beam search.
2. **Sequential search:** Sequential forward feature selection (SFFS), sequential backward feature elimination (SBFE) and bidirectional selection are greedy search algorithms that add or remove features one at a time [2]. SFFS initiates with an empty set and SBFE initiates with a full set whereas a bidirectional search initiates a SFFS and SBFE simultaneously ensuring that the features selected by SFFS are never eliminated by SBFE. The drawback of SFFS is that it could fail to eliminate redundant features generated in the search process. SBFE has the drawback of not being able to re-evaluate feature usefulness together with other features once



a feature is removed. Plus-L minus-R selection (LRS) search attempts to resolve these issues. This can be understood by the sequential forward search algorithm, which is a simple greedy search, as described by the following pseudo code.

---

**Algorithm 3** SFFS algorithm
 

---

- 1: Start with the empty set  $Y_0 = \{\emptyset\}$
  - 2: Select the next best feature  $x^+ = \arg \max x \in Y_k J(Y_k + x^+)$
  - 3: Update  $Y_{k+1} = Y_k + x^+ ; k = k + 1$
  - 4: Go to 2
- 

In the worst case every feature might be selected in which case there will be  $N$  features to select from in the first step,  $N - 1$  in the second step and so on. Therefore there will be a total of  $\sum_{i=1}^N i$  feature assessments in the worst case, and is hence  $\mathcal{O}(N^2)$ .

3. **Randomized search:** In practical applications, the feature space may contain thousands of features, e.g. bioinformatics, text, natural language processing applications and the case studies in this brief. In such applications, it is not feasible to search the entire feature space. Randomized search trades off optimality of the solution for efficiency by searching only a sampled portion of the feature space. GAs have been used as a guided randomized FS technique [11, 12] and is discussed in depth in Sect. 3.2.

Table 2.1 compares the exhaustive, sequential and randomized feature search techniques in terms of time complexity, advantages and disadvantages.

### 2.2.3 Embedded Models

In contrast to filter and wrapper models, embedded models do not separate learning from FS. For example, support vector machines (SVMs) that inherently incorporate a learning model of loss plus penalty can be equipped with embedded feature selection when a  $L_1$  norm on the weight vector is considered. This is explained below.

**Table 2.1** Search criterion comparison

Method	Complexity	Advantages	Disadvantages
Exhaustive	$\mathcal{O}(2^N)$	High accuracy	Computationally expensive
Sequential	$\mathcal{O}(N^2)$	Simple	Less flexible with backtracking
Randomized	$\mathcal{O}(N \log N)$	Users can select between accuracy and speed, avoids being trapped in local optima	Low accuracy

For a given training data set of  $(x_i, y_i)$  where the input  $x_i \in \mathbb{R}$  and the output  $y_i \in \{-1, +1\}$ , the SVM finds a hyper-plane that separates the two classes by maximizing the distance to the closest point from either class. For a problem where the two classes cannot be cleanly separated, this search for the optimal hyperplane can be viewed as the optimization problem:  $\max_{\beta_0, \beta_1} \frac{1}{\|\beta\|_2^2}$  subject to,  $y_i(\beta_0 + x_i^T \beta) \geq 1 - \xi_i, \xi_i \geq 0, \sum_{i=1}^n \xi_i \leq B$  where  $\xi$ 's are the slack variables associated with the overlap between the two classes,  $B$  is the tuning parameter that is responsible for controlling this overlap and  $\beta_0, \beta$  are the decision boundary variables. Although the problem shown is for binary classification, SVM has also been extended to multi-class classification and regression problems.

The optimization problem above is equivalent to the standard SVM model with a  $L_2$  norm in the form of loss + penalty,

$$\min_{\beta_0, \beta_j} \left[ \sum_{i=1}^n 1 - y_i \left( \beta_0 + \sum_{j=1}^q \beta_j h_j(x_i) \right) \right] + \lambda \|\beta\|_2^2 \quad (2.8)$$

where  $\lambda$  is the parameter which balances the tradeoff between the loss and penalty,  $n$  is the number of training samples and  $q$  is the dimensionality of the solution space  $\mathbb{R}^q$ . The classification rule is given by  $(\beta_0 + x^T \beta)$ . The  $h_j(x)$ s are usually chosen to be the basis functions of a reproducing kernel Hilbert space (RKHS) which allows the use of the kernel trick to transform the original feature space to a higher dimension [13]. There are specific conditions that should be satisfied in choosing a kernel such as the inner product being positive semi-definite, but we omit the details here. The loss term in the above equations is called the hinge loss, and the penalty is called the ridge penalty which is also used in ridge regression. The ridge penalty shrinks the fitted coefficients  $\beta$ 's to approach zero.

Zhu et al. [13] proposed to use the  $L_1$  norm, instead of the standard  $L_2$  norm which will give the equivalent Lagrange version of the optimization problem,

$$\min_{\beta_0, \beta_j} \left[ \sum_{i=1}^n 1 - y_i \left( \beta_0 + \sum_{j=1}^q \beta_j h_j(x_i) \right) \right] + \lambda \|\beta\|_1 \quad (2.9)$$

This is known as the lasso penalty which also shrinks  $\beta$ 's towards zero. However, by making  $\lambda$  sufficiently large, some of the coefficients  $\beta$  can be made exactly zero. Therefore, the lasso penalty introduces a better form of integrated feature selection that is not observed when using the ridge penalty.

Ng [14] draws the following conclusions in regularization using  $L_1$  and  $L_2$  norms. (i)  $L_1$  outperforms  $L_2$  regularization for logistic regression when there are more irrelevant dimensions than training examples, e.g.  $n = 100$ , and a large number of basis functions, e.g.  $q = 1,000$  (translates to features). This is attributed to the fact that in a sparse scenario, only a small number of true coefficients ( $\beta_j$ s) are non-zero hence the lasso penalty works better than the ridge penalty. (ii)  $L_2$  regularization classifies

poorly for even a few irrelevant features (iii) Poor performance of  $L_2$  regularization is linked to rotational invariance and (iv) Rotational invariance is shared by a large class of other learning algorithms. These other algorithms presumably have similarly bad performance with many irrelevant dimensions, especially when there are many features than samples. SVM-RFE (Recursive Feature Elimination) is an embedded FS method proposed in [14] for gene selection in cancer classification. If a user wishes to select a subset of only  $m$  features, nested subsets of features are selected in a sequential backward elimination manner, which starts with all the features and removes one or more feature at a time (greedy backward selection). At each iteration, the feature that decreases the margin the least will be eliminated until only  $m$  features remain. Many other embedded feature selection techniques for different learning algorithms exist but we have specifically focused on SVM embedded models in this brief.

**Ensemble Learning** Ensemble methods combine multiple learning algorithms, and numerous studies show that they can improve the performance over a single learner. There are two variations of ensemble learning; parallel and serial [15]. A parallel ensemble method combines independently constructed learners from the same dataset, e.g. each learner is trained on a different bootstrap sample (a sample drawn with replacement) from the training dataset. When the learners show diverse performance, i.e. different learners show different errors on unseen data, errors cancel out to produce better results than a single learner. For example, random forests (RFs) are a parallel ensemble technique which reduces error variance. In serial ensembles, the new learner is based on the previously built learner, i.e. the final model is built in a forward stage by stage pattern, e.g. Adaboost, Gradient tree boosting. For example, in Adaboost, a sequence of weak learners i.e., models that are only slightly better than random guessing, such as small decision trees or decision stumps are trained on repeatedly reweighted data samples.

## References

1. R.Kohavi, G.H. John, Wrappers for feature subset selection. *Artif. intell.* **97**(1), 273–324 (1997)
2. H. Liu, Y. Lei, Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **17**(4), 491–502 (2005)
3. I. Guyon, An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
4. L.J. Cao, K.S. Chua, W.K. Chong, H.P. Lee, Q.M. Gu, A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. *Neurocomputing* **55**(1), 321–336 (2003)
5. A. Jakulin, I. Bratko, Testing the significance of attribute interactions. in *Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04* (ACM, New York, 2004), p. 52
6. S. K. Singhi, H. Liu, Feature subset selection bias for classification learning. in *Proceedings of the 23rd International Conference on Machine Learning* (ACM, New York, 2006), pp. 849–856
7. I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh, Feature extraction, *Foundation and applications* (Springer, Science and Business Media, 2006)

8. K. Kira, L.A. Rendell, A practical approach to feature selection. in *Proceedings of the 9th International Workshop on Machine Learning, ML92* (Morgan Kaufmann Publishers Inc., San Francisco, 1992), pp. 249–256
9. M. Dash, Feature selection via set cover. in *Knowledge and Data Engineering Exchange Workshop, 1997. Proceedings*, pp. 165–171 (1997)
10. H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Anal. Mach. Intell., IEEE Trans.* 27(8):1226–1238 (2005)
11. K.J. Cherkauer, J.W. Shavlik, Growing simpler decision trees to facilitate knowledge discovery, in *KDD'96*, pp. 315–318 (1996)
12. H. Vafaie, K. De Jong, Genetic algorithms as a tool for restructuring feature space representations. in *Proceedings of 7th International Conference on Tools with Artificial Intelligence*, pp. 8–11 (1995)
13. J. Zhu, S. Rosset, T. Hastie, R. Tibshirani, 1-norm support vector machines. *Adv. Neural Inf. Process. Syst.* 16(1), 49–56 (2004)
14. A.Y. Ng, Feature selection,  $l_1$  versus  $l_2$  regularization, and rotational invariance, in *Proceedings of the Twenty-first International Conference on Machine Learning* (ACM, New York, 2004), p. 78
15. I. Guyon, A. Elisseeff, An introduction to variable and feature selection. *J. Mach. Learn. Res.* 3, 1157–1182 (2003)

Grammar-Based Feature Generation for Time-Series  
Prediction

De Silva, A.M.; Leong, P.H.W.

2015, XI, 99 p. 28 illus., Softcover

ISBN: 978-981-287-410-8