# Combinatorial Optimization in Data Mining

Samira Saedi and O. Erhun Kundakcioglu*

## Contents



*This work is supported by University of Houston New Faculty Research Grant.

S. Saedi (✉) • O.E. Kundakcioglu
Department of Industrial Engineering, University of Houston, Houston, TX, USA
e-mail: ssaedi@uh.edu; erhun@uh.edu

P.M. Pardalos et al. (eds.), *Handbook of Combinatorial Optimization*,
DOI 10.1007/978-1-4419-7997-1_7, © Springer Science+Business Media New York 2013

**Abstract**

This chapter presents data mining techniques that are formulated as combinatorial optimization problems together with their applications. There are a number of cases where fundamental data mining tool is not combinatorial in nature, yet widely used special-purpose combinatorial extensions exist. For the sake of completeness, these fundamental tools are also discussed in detail before the extensions with underlying combinatorial optimization problems. A number of computationally challenging data mining algorithms that have non-convex formulations are also explored.

## 1 Introduction

Data mining is defined as the practice of searching through large amounts of computerized data to find useful patterns or trends [101]. Being closely related to statistics and machine learning, data mining has been referred to as statistical learning and learning from the data. Different classifications of data mining methods exist in the literature [24], but one of the most commonly used classifications is based on the type of input data. Other than the features (attributes), input data can contain information on classes data instances belong to or responses of the underlying system, or it may contain no additional prior information. If information on classes or responses are available, the data is considered as *labeled*, and *supervised learning techniques* would be appropriate. Conversely, *unlabeled* data consists of only features with no class or response information, where *unsupervised learning techniques* attempt to identify patterns in data. *Semi-supervised learning*, which can be considered as halfway between supervised and unsupervised learning, utilizes both labeled and unlabeled data.

Regardless of the data mining tool employed, a data instance belongs to one of the two sets: *training set* or *test set*. Training set contains data instances that are used to train the data mining tool. That includes discovering patterns and relationships. Test set contains data instances that are used to assess the success of the predictions on relationships, which shed a light on the generalization performance of the employed tool.

Generalization performance of a model is its capability to find a more general classification based on given instances, i.e., how accurate the model can classify independent sample sets. To evaluate the generalization performance of a mining model, *cross validation* methods are used. Two widely used cross validation methods are *k-fold cross validation* and *leave-one-out cross validation (jackknife)*. In $k$-fold cross validation, all available data is randomly partitioned into $k$ approximately equal subsets. Then, one of the $k$ subsets is chosen as test set and the rest of the subsets are used as training set. This process is repeated $k$ times in a way that each subset is chosen as test set once. The $k$ results obtained will be averaged to find the generalization performance of the classification method. Leave-one-out cross validation (LOOCV), also known as jackknife, is a special case of $k$-fold where $k$ is equal to the number of data instances.

In this chapter, methods to learn from data are explored in three main sections: supervised, unsupervised, and semi-supervised, with an emphasis on underlying combinatorial optimization problems. Discrete decision making based on analysis of data is the goal of fundamental problems in data mining (e.g., class assignment, feature selection, data categorization, identifying outlier instances). These problems are combinatorial in nature and can be formulated and solved by combinatorial approaches [33]. Last two sections are devoted to combinatorial ranking and feature selection problems.

## 2 Supervised Learning

Supervised learning refers to data mining tools that utilize *labeled* data. These labels can be *discrete* categories or *continuous* responses where the corresponding techniques are called *classification* or *regression*, respectively. Labels are input to the data mining tool during the training stage by a *supervisor*, hence the name supervised learning. In this section, combinatorial optimization problems are investigated in the context of classification and regression.

### 2.1 Support Vector Machines

*Support vector machines* (SVMs) are the state-of-the-art supervised machine learning methods that are initially introduced to classify pattern vectors that belong to two different classes (see [136]). Although there are multi-class generalizations, hyperplane-based methods have major drawbacks in classifying data from multiple classes (see [19]). Besides, classification of two classes has a wide variety of applications in image and voice recognition, text mining, healthcare, and clinical data mining.

The SVM classification function is a hyperplane that separates given two classes. The desired hyperplane maximizes the distance from the convex hulls of both classes. This problem can be formulated as a quadratic (and convex) optimization problem. SVM classifiers' success relies on strong fundamentals from the statistical learning theory, implementation advantages due to regularization (hence sparsity), and their generalization performance. When misclassified instances are penalized in the linear form, SVM classifiers are proven to be universally consistent (see [130]). A classifier is *consistent* if the probability of misclassification (in expectation) converges to a Bayes optimal rule when the number of data instances increases. A classifier is *universally consistent* if it is consistent for all distributions of data. SVMs can also perform nonlinear classification utilizing separating curves by implicitly embedding the original data in a nonlinear space using *kernel functions* (see, e.g., [121]).

The training is performed by minimizing a quadratic convex function that is subject to linear constraints. Although minimizing a convex function has a polynomial worst-case complexity, the general purpose methods are not practical for

large problems. SVM Light [78] and LIBSVM [71] are among the most frequently used implementations that use chunking [107] and decomposition [110] methods efficiently and use subsets of points to find a near-optimal hyperplane. Recently, Shalev-Shwartz et al. [120] propose a stochastic sub-gradient descent algorithm (Pegasos), whose run-time does not depend directly on the size of the training set but the bound on the number of nonzero features in each data instance. Experimental results show that Pegasos is especially successful for linear kernels.

SVM classifiers have a wide spectrum of application areas ranging from pattern recognition [92] and text categorization [77] to biomedicine [26, 43, 106, 109], cell death [112], nanotoxicology [113], brain-computer interface [60, 89], and financial applications [72, 134].

Based on SVM classifiers, support vector regression (SVR) is an optimization-based regression framework for solving machine learning problems. SVR approach is based on estimation of a linear function in a kernel-induced feature space. The objective is to optimize a certain boundary to the optimal regression line; therefore, errors within a certain distance ($\varepsilon$) of predicted value are disregarded. The learning algorithm minimizes a convex functional with sparse solution comparable to classification technique. For improved illustration, this can be considered a hyper-tube (insensitive band) about a linear function in the kernel-induced nonlinear space, such that pattern vectors in this tube are assumed not to contribute any error. This form of regression is called $\varepsilon$-insensitive because any point within $\varepsilon$ distance of the anticipated regression function does not contribute an error. An important advantage for considering the $\varepsilon$-insensitive loss function is the sparseness of the dual variables similar to the case with SVM classifiers. Representing the solution by a small subset of training points has computational advantages. Furthermore, $\varepsilon$-insensitive regression ensures the existence of a global minimum and minimization of a reliable generalization error bound (see [45]).

SVR has various applications in numerous technological [15, 116], analytical [73, 91], and scientific fields [131, 145]. Wu et al. [142] perform location estimation using the Global System for Mobile communication (GSM) based on an SVR approach which demonstrates promising performances, especially in terrains with local variations in environmental factors. SVR method is also used in agricultural schemes in order to enhance output production and reduce losses [42, 108, 143]. Based on statistical learning theory, SVR has been used to deal with forecasting problems. Performing structural risk minimization rather than minimizing the training errors, SVR algorithms have better generalization ability than the conventional artificial neural networks [70].

### 2.1.1 SVM Classifiers: Mathematical Formulation

In a typical *binary classification* problem, class $\mathbf{S}^+$ and $\mathbf{S}^-$ are composed of pattern vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$. If $\mathbf{x}_i \in \mathbf{S}^+$, it is given the label $y_i = 1$; if $\mathbf{x}_i \in \mathbf{S}^-$, then it is given the label $y_i = -1$. The ultimate goal is to determine which class a new pattern vector $\mathbf{x}_i \notin \{\mathbf{S}^+ \cup \mathbf{S}^-\}$ belongs to. SVM classifiers solve this problem by finding a hyperplane $(\mathbf{w}, b)$ that separates instances in classes $\mathbf{S}^+$ and $\mathbf{S}^-$ with the maximum interclass margin.

A hyperplane in the feature space $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ is represented as the normal vector $\mathbf{w}$ and the offset parameter $b$. The *geometric distance* to be maximized between a data point $\mathbf{x}_i$ and the hyperplane is given by $(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)/\|\mathbf{w}\|$. Instead of maximizing the geometric distance fixing $\|\mathbf{w}\|$, the approach is minimizing $\|\mathbf{w}\|$ fixing $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$, which is referred to as the *functional distance*.

In practice, many real-life problems are composed of nonseparable data which is generally due to noise. In this case, *slack variables* $\xi_i$ are introduced for each pattern vector $\mathbf{x}_i$ in the training set. Slack variables allow misclassifications for each pattern vector, but they are subject to a penalty to avoid trivial solutions. Therefore, the SVM classifier formulation is given as

$$[\mathbf{SVM}] \quad \min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n} \xi_i^2 \tag{1a}$$

$$\text{subject to} \quad y_i\left(\langle \mathbf{w}, \mathbf{x_i} \rangle + b\right) \geq 1 - \xi_i \qquad i = 1, \ldots, n, \tag{1b}$$

where nonnegativity of the slack variables is assured implicitly since the solution cannot be optimal when $\xi_i < 0$ for any pattern vector.

Using the optimal solution $(\mathbf{w}^*, b^*)$ for (1), a new pattern vector $\mathbf{x}'$ can be classified as positive if $\langle \mathbf{w}^*, \mathbf{x}' \rangle + b^* > 0$ and negative if $\langle \mathbf{w}^*, \mathbf{x}' \rangle + b^* < 0$ (see Fig. 1). It is common to penalize the two-norm of the slack in the objective of SVM classifiers. Alternative formulations exist with one-norm penalization of the slack vector in the objective function, which is commonly referred to as the *hinge loss function* [45].

*Lagrangian dual* formulation of (1) and optimality conditions lead to an optimization problem where input vectors only appear in the form of dot products. Therefore, *kernel trick* can be introduced for nonlinear classification [45]. The dual problem is a concave maximization problem, which can also be solved efficiently. However, when the number of data points increases for better generalization performance, the dual tends to get harder to solve compared to the primal. The dual for two-norm soft margin formulation in (1) is given as

$$[\mathbf{Dual-SVM}] \quad \max_{\boldsymbol{\alpha}} \ \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \frac{1}{2C}\sum_{i=1}^{n} \alpha_i^2 \tag{2a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{2b}$$

$$\alpha_i \geq 0 \qquad i = 1, \ldots, n. \tag{2c}$$

When the dual formulation is used, $b^*$ is calculated as follows using Karush-Kuhn-Tucker complementarity conditions:

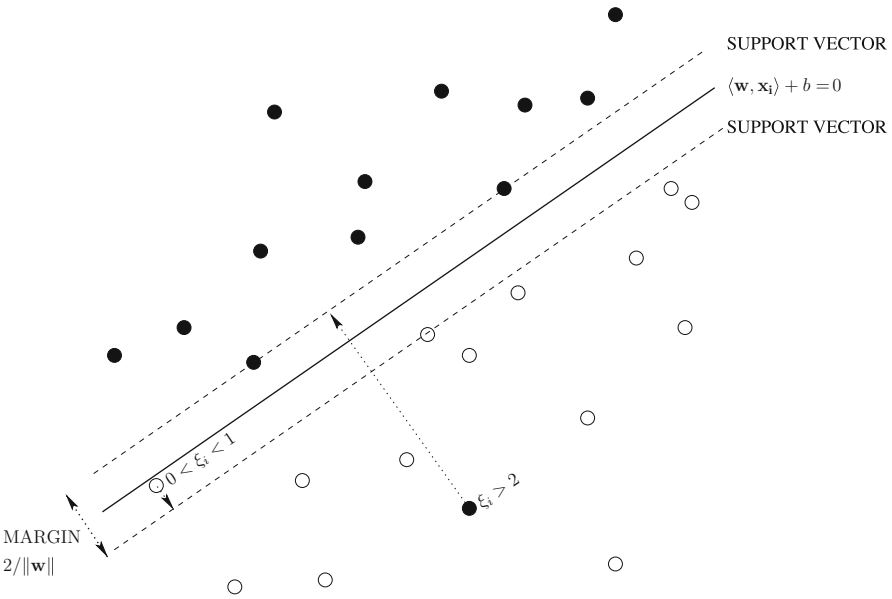


**Fig. 1** Illustration of linear support vector machine classifiers

$$b^* = \sum_{i:\alpha_i^* > 0} y_i - \sum_{j=1}^{n} y_j \alpha_j^* \langle \mathbf{x}_j, \mathbf{x}_i \rangle, \tag{3}$$

and a new pattern vector, $\mathbf{x}'$, can be classified as positive if $\sum_{j=1}^{n} y_i \alpha_j^* \langle \mathbf{x}_j, \mathbf{x}' \rangle + b^* > 0$, and negative otherwise. Note that dot products in (2) and (3) can be substituted with a suitable kernel function. The kernel function transforms the original *input space*, $\mathcal{X}$ to a usually higher dimensional dot product space $\mathcal{H}$ called the *feature space*, with a map $\mathbf{\Phi} : \mathcal{X} \rightarrow \mathcal{H}$, such that $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle$. The function must satisfy conditions for Mercer's theorem which are equivalent to the requirement that the corresponding matrix is positive semidefinite for any finite subset of $\mathcal{X}$. Further information on the kernel trick can be found in [45, 117].

### 2.1.2 SVM Regressors: Mathematical Formulation

There are many reasonable choices of loss function for regression. SVR uses $\varepsilon$-insensitive loss function to ensure that the solution is characterized as the minimum of a convex functional. Another motivation for considering the $\varepsilon$-insensitive loss function is that it will ensure sparseness of the dual variables similar to SVM classifiers.

The linear $\varepsilon$-insensitive loss function $L^{\varepsilon}(\mathbf{x}, y, f)$ is defined by

$$L^{\varepsilon}(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_{\varepsilon} = \max(0, |y - f(\mathbf{x})| - \varepsilon),$$

where $f$ is a real-valued function on a domain $X$, $\mathbf{x} \in X$, and $y \in \mathbb{R}$. The quadratic $\varepsilon$-insensitive loss function $L_2^{\varepsilon}(\mathbf{x}, y, f)$ is similarly

$$L_2^{\varepsilon}(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_{\varepsilon}^2.$$

Let $\mathbf{X}$ be a set of pattern vectors $\mathbf{x}_i \in \mathbb{R}^d$, with dependent variable values (i.e., real-valued response) $y_i \in \mathbb{R}$. The problem of finding a regression function $f(\cdot)$ for these pattern vectors is minimizing the sum of $\varepsilon$-insensitive losses over all pattern vectors $\mathbf{x}$. Let the distance between the regression hyperplane and the closest pattern vector $\mathbf{x}^*$ be $|(\langle \mathbf{w}, \mathbf{x}^* \rangle) + b|$. Then, the solution to the following quadratic programming problem finds the regression hyperplane with the minimum sum of quadratic $\varepsilon$-insensitive losses. Similar to the goal programming approach in SVM classifiers, there is an associated penalty of $C$ for outliers. Therefore, the primal SVR problem is as follows:

$$[\mathbf{SVR}] \quad \min_{\mathbf{w},b,\boldsymbol{\xi},\hat{\boldsymbol{\xi}}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\left(\xi_i^2 + \hat{\xi}_i^2\right) \tag{4a}$$

$$\text{subject to} \quad (\langle \mathbf{w}, \mathbf{x_i} \rangle + b) - y_i \leq \varepsilon + \xi_i \qquad i = 1, \ldots, n \tag{4b}$$

$$y_i - (\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \leq \varepsilon + \hat{\xi}_i \qquad i = 1, \ldots, n. \tag{4c}$$

The main idea is to create a linear function in the kernel-induced space such that the quadratic loss function for regression from the generalization theory is minimized. First component of the objective function deals with minimizing the squared norm of the regression hyperplane ($\|\mathbf{w}\|^2$). The goal here is to enclose all pattern vectors with the $\varepsilon$-insensitive band of the regression hyperplane. Second component of the objective introduces penalty cost for instances outside the $\varepsilon$ boundaries. Note that both $\xi_i$ and $\hat{\xi}_i$ will be zero for all points inside the limits. The objective function seeks to minimize this penalty cost to reveal the best regression fit to the model. Constraints (4b)–(4c) imply that the pattern vectors are allowed to be $\varepsilon$ below or above the target value without penalty. All pattern vectors outside the $\varepsilon$ range are still allowed; however, they incur a cost of $C$ [121].

The dual can be found using the Lagrangian function for the primal problem, differentiating this function with respect to the primal variables, and substituting equivalent expression for the primal variables back in the Lagrangian function. The resulting dual formulation is given as

$$[\mathbf{Dual-SVR}] \quad \max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_i - \alpha_i)\langle \mathbf{x_i}, \mathbf{x_j} \rangle \tag{5a}$$

$$-\varepsilon\sum_{i=1}^{n}(\alpha_i + \hat{\alpha}_i) + \sum_{i=1}^{n} y_i(\hat{\alpha}_i - \alpha_i)$$

$$\text{subject to} \quad \sum_{i=1}^{n}(\alpha_i - \hat{\alpha}_i) = 0 \tag{5b}$$

$$0 \leq \alpha_i, \hat{\alpha}_i \leq C \qquad i = 1, \ldots, n. \tag{5c}$$

From the solution $\boldsymbol{\alpha}^*$ and $\hat{\boldsymbol{\alpha}}^*$, the regression function can be written as $f(\mathbf{x}) = \sum_{i=1}^{n}(\hat{\alpha}_i^* - \alpha_i^*)\langle \mathbf{x}_i, \mathbf{x}\rangle + b^*$, where $b^*$ is chosen such that $f(\mathbf{x}_i) - y_i = -\varepsilon$ for any $i$ with $0 < \hat{\alpha}_i^* < C$.

### 2.1.3 Combinatorial Extensions of Support Vector Machines

Recently proposed combinatorial optimization problems related to SVMs are based on penalization of different loss functions and generalizations of traditional classification problem. Integer programming formulations for SVM classifiers with the ramp loss (6) and hard margin loss (7) are proposed in [25]. Facet-defining inequalities are presented and both ramp loss and hard margin loss SVM classifiers are proven to be universally consistent. The idea behind these combinatorial formulations is to obtain classifiers that are more robust to the outliers compared to hinge or smooth loss functions:

$$[\mathbf{RL-SVM}] \quad \min_{\mathbf{w},b,\boldsymbol{\xi},\mathbf{z}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n}\xi_i + 2\sum_{i=1}^{n} z_i\right) \tag{6a}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 - \xi_i \qquad \text{if } z_i = 0, \qquad i = 1, \ldots, n \tag{6b}$$

$$z_i \in \{0, 1\} \qquad i = 1, \ldots, n \tag{6c}$$

$$0 \leq \xi_i \leq 2 \qquad i = 1, \ldots, n \tag{6d}$$

$$[\mathbf{HML-SVM}] \quad \min_{\mathbf{w},b,\mathbf{z}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} z_i \tag{7a}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 \qquad \text{if } z_i = 0, i = 1, \ldots, n \tag{7b}$$

$$z_i \in \{0, 1\} \qquad i = 1, \ldots, n. \tag{7c}$$

Seref et al. [118] introduce novel *selective* linear and nonlinear classification methods, in which sets of pattern vectors sharing the same label are given as input. One pattern vector is *selected* from each set in order to maximize the classification margin with respect to the selected positive and negative pattern vectors. The problem of selecting the best pattern vectors is referred to as the *hard selection* problem. Kernelized hard selection problems are also developed for classification. However, these combinatorial problems cannot be solved in polynomial time unless $\mathcal{P} = \mathcal{NP}$ [119]. Alternative approaches are proposed with relaxed formulations. The selective nature of these formulations is satisfied by the restricted free slack

concept. The intuition behind this concept is to reverse the combinatorial selection problem by detecting influential pattern vectors which require free slack to decrease their effect on the classification functions. Iteratively removing problematic pattern vectors, better classification results can be found.

Two variations of the free slack method, namely, pooled free slack (PFS) and free slack per set (FSS), are introduced for selective linear classification together with kernelized dual formulations for selective nonlinear classification. These methods are further extended to direct separation by increasing the total free slack to diminish the effect of multiple pattern vectors per set and provide more flexibility for the hyperplane to reorient itself with respect to well-separated pattern vectors. The performance of iterative elimination and direct selection algorithms is compared with each other, as well as with a naïve elimination algorithm that uses standard SVM method and ideas from the proposed methods. Results are reported for linear and nonlinear simulated data.

Kundakcioglu et al. [88] consider the margin maximization problem within the multiple instance learning (MIL) context. Training data is composed of labeled bags of instances. Despite the large number of margin maximization-based classification methods, there are only a few methods that consider the margin for MIL problems in the literature. A combinatorial margin maximization problem (8) is formulated for multiple instance classification which is proven to be $\mathcal{NP}$-hard. Kernel trick is applied on this formulation to classify nonlinear MIL data. A branch and bound algorithm is proposed that outperforms a leading commercial solver in terms of the best integer solution and optimality gap in a majority of image annotation and molecular activity prediction test cases. The major difference between the MIL setting and the selective setting is the interpretation of negative bags. In selective learning, a selection is performed on negative bags as well as positive bags. In MIL, on the other hand, only actual positives are to be discovered where all negative instances must be kept. The mixed-integer nonlinear programming (MINLP) formulation for this problem is as follows:

$$
\begin{aligned}
[\mathbf{MI-SVM}] \quad \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\eta}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 && && \text{(8a)}\\
\text{s.t.} \quad & \langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b \geq 1-\xi_i - M(1-\eta_i) && i \in I^+ && \text{(8b)}\\
& -\langle \mathbf{w}\cdot\mathbf{x}_i\rangle - b \geq 1-\xi_i && i \in I^- && \text{(8c)}\\
& \sum_{i\in I_j}\eta_i \geq 1 && j \in J^+ && \text{(8d)}\\
& \eta_i \in \{0,1\} && i \in I^+. && \text{(8e)}
\end{aligned}
$$

In this formulation, $I^+ = \{i : i \in I_j \wedge y_j = 1\}$ is the index set for instances that are in a positive bag, $I^- = \{i : i \in I_j \wedge y_j = -1\}$ is the index set for instances that are in a negative bag (negative instances), and $J^+ = \{j : y_j = 1\}$ is the index set for positive bags. Note that $M$ is a sufficiently large number that ensures that

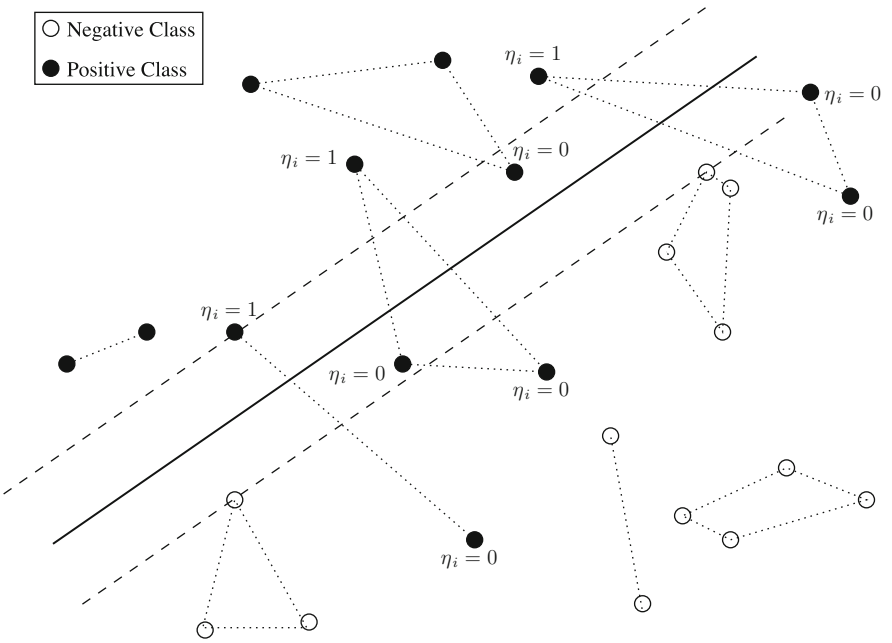


**Fig. 2** Multiple instance support vector machine classifiers

the constraint is active if and only if $\eta_i = 1$. $\eta_i$ is a binary variable that is 1 if $i$th instance is one of the actual positive examples of its bag (see Fig. 2).

Recently, Poursaeidi and Kundakcioglu [111] propose a hard margin loss formulation for multiple instance learning. The main idea is to avoid the dependency of the classifier in (Eq. 8) to the number of negative instances and make it dependent on the number of negative bags. Proposed formulations in [111] are also less sensitive to outliers, showing better generalization performance.

Next, *consistent biclustering* is introduced, another classification technique that employs combinatorial optimization formulations.

## 2.2 Consistent Biclustering

The concept of *consistent biclustering* is introduced by Busygin et al. [29]. In this method, a classification of features as well as data instances are needed as input. Formally, a biclustering $\mathcal{B}$ is consistent if in each instance (feature) from any set $S_r$ (set $F_r$), the average expression of features (instances) that belong to the same class $r$ is greater than the average expression of features (instances) from other classes. The model for supervised biclustering involves solution of a special case of fractional 0–1 programming problem whose consistency is achieved by feature

selection. Computational results on microarray data mining problems are obtained by reformulating the problem as a linear mixed 0–1 programming problem.

An improved heuristic procedure is proposed in [103], where a linear programming problem with continuous variables is solved at each iteration. Numerical experiments on the data, which consists of instances from patients diagnosed with *acute lymphoblastic leukemia (ALL)* or *acute myeloid leukemia (AML)* diseases (see [10,11,64,141,144]), confirm that the algorithm outperforms the previous results in the quality of solution as well as computation time. Busygin et al. [30] use consistent biclustering to analyze scalp EEG data obtained from epileptic patients undergoing treatment with a vagus nerve stimulator (VNS).

Given an $m \times n$ data matrix $A$, each column represents a data instance and each row represents a feature. Formally, $A = (a_{ij})_{m\times n}$, where $a_{ij}$ is the expression of $i$th feature of $j$th instance. A classification of instances is provided through a 0–1 matrix $S = (s_{jr})_{n\times k}$, where $s_{jr} = 1$ if instance $j$ is classified as a member of the class $r$ (i.e., $a^j \in S_r$), and $s_{jr} = 0$ otherwise. Similarly, given a classification of the features, $F_r$, let $F = (f_{ir})_{m\times k}$ denote a 0–1 matrix where $f_{ir} = 1$ if feature $i$ belongs to class $r$ (i.e., $a_i \in F_r$), and $f_{ir} = 0$ otherwise. Corresponding *centroids* are constructed as follows:

$$C_S = AS(S^T S)^{-1} = (c^S_{i\xi})_{m\times r}, \tag{9}$$

$$C_F = A^T F(F^T F)^{-1} = (c^F_{j\xi})_{n\times r}. \tag{10}$$

The elements of the matrices, $c^S_{i\xi}$ and $c^F_{j\xi}$, represent the average expression of the corresponding instance and feature in class $\xi$, respectively:

$$c^S_{i\xi} = \frac{\sum_{j=1}^n a_{ij}s_{j\xi}}{\sum_{j=1}^n s_{j\xi}} = \frac{\sum_{j\,|a^j\in S_\xi} a_{ij}}{|S_\xi|},$$

and

$$c^F_{j\xi} = \frac{\sum_{i=1}^m a_{ij}f_{i\xi}}{\sum_{i=1}^m f_{i\xi}} = \frac{\sum_{i\,|a_i\in F_\xi} a_{ij}}{|F_\xi|}.$$

Using the elements of matrix $C_s$, one can assign a feature to a class where it is overexpressed. Therefore, feature $i$ is assigned to class $\hat{r}$ if $c^S_{i\hat{r}} = \max_\xi\{c^S_{i\xi}\}$. Similarly, one can use the elements of matrix $C_F$ to classify the instances. Data instance $j$ is assigned to class $\hat{r}$ if $c^F_{j\hat{r}} = \max_\xi\{c^F_{j\xi}\}$. Formally,

$$a_i \in \hat{F}_{\hat{r}} \Longrightarrow c^S_{i\hat{r}} > c^S_{i\xi}, \qquad \forall\xi, \xi \neq \hat{r}, \tag{11}$$

$$a^j \in \hat{S}_{\hat{r}} \Longrightarrow c^F_{j\hat{r}} > c^F_{j\xi}, \qquad \forall\xi, \xi \neq \hat{r}. \tag{12}$$

Note that the constructed classification of the features and instances $\hat{F}_r$ and $\hat{S}_r$ are not necessarily the same as classifications $F_r$ and $S_r$, respectively.

Biclustering $\mathcal{B}$ is referred to as a *consistent biclustering* if relations (11) and (12) hold for all elements of the corresponding classes, where matrices $C_S$ and $C_F$ are defined according to (9) and (10), respectively. A data set is *biclustering-admitting* if some consistent biclustering for it exists. Furthermore, the data set is called *conditionally biclustering-admitting* with respect to a given (partial) classification of some instances and/or features if there exists a consistent biclustering preserving the given (partial) classification. Busygin et al. [29] prove conic separability which also indicates convex hulls of classes do not intersect.

By definition, a biclustering is consistent if $F_r = \hat{F}_r$ and $S_r = \hat{S}_r$. However, a given data set might not have these properties. In such cases, one can remove a set of features and/or instances from the data set so that there is a consistent biclustering for the truncated data. This feature selection process may incorporate various objective functions depending on the desirable properties of the selected features. One general choice is to select the maximal possible number of features in order to lose minimal amount of information provided by the training set:

$$[\mathbf{CB}] \quad \max_{\mathbf{x}} \quad \sum_{i=1}^{m} x_i \tag{13a}$$

$$\text{subject to} \quad \frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{r}} x_i}{\sum_{i=1}^{m} f_{i\hat{r}} x_i} > \frac{\sum_{i=1}^{m} a_{ij} f_{i\xi} x_i}{\sum_{i=1}^{m} f_{i\xi} x_i} \quad \hat{r}, \xi = 1, \ldots, k, \hat{r} \neq \xi, j \in S_{\hat{r}} \tag{13b}$$

$$x_i \in \{0, 1\} \quad i = 1, \ldots, m. \tag{13c}$$

In this formulation, $x_i$, $i = 1, \ldots m$ are the decision variables. $x_i = 1$ if $i$th feature is selected, and $x_i = 0$ otherwise. $f_{ik} = 1$ if feature $i$ belongs to class $k$, and $f_{ik} = 0$ otherwise. The objective is to maximize the number of features selected and (13b) ensures that the biclustering is consistent with respect to the selected features.

The goal of the CB problem is to find the largest set of features that can be used to construct a consistent biclustering. A problem with selecting the most representative feature set is the following. Assume that there is a consistent biclustering for a given data set, and there is a feature, $i$, where the difference between the two largest values of $c^S_{ir}$ is negligible. For such cases, *additive* and *multiplicative consistent biclustering* are introduced in [103] by relaxing (11)–(12) with (14)–(15) and (16)–(17), respectively:

$$a_i \in F_{\hat{r}} \Longrightarrow c^S_{i\hat{r}} > \alpha^S_i + c^S_{i\xi}, \qquad \forall \xi, \xi \neq \hat{r} \tag{14}$$

$$a^j \in S_{\hat{r}} \Longrightarrow c^F_{j\hat{r}} > \alpha^F_j + c^F_{j\xi}, \qquad \forall \xi, \xi \neq \hat{r} \tag{15}$$

$$a_i \in F_{\hat{r}} \Longrightarrow c^S_{i\hat{r}} > \beta^S_i c^S_{i\xi}, \qquad \forall \xi, \xi \neq \hat{r} \tag{16}$$

$$a^j \in S_{\hat{r}} \Longrightarrow c^F_{j\hat{r}} > \beta^F_j c^F_{j\xi}, \qquad \forall \xi, \xi \neq \hat{r}. \tag{17}$$

Using these relaxations, $\alpha$-consistent and $\beta$-consistent biclustering problem formulations are obtained as follows:

$$[\alpha\text{–}\mathbf{CB}] \quad \max_{\mathbf{x}} \quad \sum_{i=1}^{m} x_i \tag{18a}$$

$$\text{subject to} \quad \frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{r}} x_i}{\sum_{i=1}^{m} f_{i\hat{r}} x_i} > \alpha_j + \frac{\sum_{i=1}^{m} a_{ij} f_{i\xi} x_i}{\sum_{i=1}^{m} f_{i\xi} x_i} \quad \forall \hat{r},$$

$$\xi = 1, \ldots, k, \hat{r} \neq \xi, j \in S_{\hat{r}} \tag{18b}$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, m \tag{18c}$$

$$[\beta\text{–}\mathbf{CB}] \quad \max_{\mathbf{x}} \quad \sum_{i=1}^{m} x_i \tag{19a}$$

$$\text{subject to} \quad \frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{r}} x_i}{\sum_{i=1}^{m} f_{i\hat{r}} x_i} > \beta_j \frac{\sum_{i=1}^{m} a_{ij} f_{i\xi} x_i}{\sum_{i=1}^{m} f_{i\xi} x_i} \quad \forall \hat{r},$$

$$\xi = 1, \ldots, k, \hat{r} \neq \xi, j \in S_{\hat{r}} \tag{19b}$$

$$x_i \in \{0, 1\} \qquad i \in 1, \ldots, m. \tag{19c}$$

The information obtained from these solutions can be used to classify additional instances in the *test set*. These solutions are also useful for adjusting the values of vectors $\alpha$ and $\beta$ to produce more characteristic features and decrease the number of misclassifications. Feature selection for consistent (13), $\alpha$-consistent (18), and $\beta$-consistent biclustering (13) is proven to be $\mathcal{NP}$-hard [86].

## 2.3 Other Classification Approaches

A special case of biclustering introduced by Ben-Dor et al. [12] is the order-preserving submatrix problem, which is proven to be $\mathcal{NP}$-hard. The goal of this problem is to select a subset of rows and columns from the original data matrix in which there exists a permutation of columns such that in each row the values are strictly increasing. Trapp and Prokopyev [135] propose a general linear mixed 0–1 programming formulation and an iterative algorithm that makes use of a smaller linear 0–1 programming formulations. The proposed solution algorithm enhanced by a number of enhancements including valid inequalities and bounding schemes is able to solve problems with approximately 1,000 rows and 50 columns to optimality.

Bertsimas and Shioda [16] introduce mixed-integer linear methods to the classical statistical problems of classification and regression and construct a software

package called *classification and regression via integer optimization* (CRIO). CRIO separates data points into different polyhedral regions. In classification, each region is assigned a class, while in regression, each region has its own distinct regression coefficients. Computational experimentations show that CRIO is comparable to the leading methods in classification and regression.

*Logical analysis of data* is a technique that is used for risk prediction in medical applications [2]. This method is based on combinatorial optimization and boolean logic. The goal is essentially classifying groups of patients at low and high mortality risk, and LAD is shown to outperform standard methods used by cardiologists.

Kundakcioglu and Ünlüyurt [87] consider the problem of generating the sequence of tests required to reach a diagnostic conclusion with minimum average cost, which is also known as a test-sequencing problem. In this setting, the training data consists of asymmetrical tests (i.e., a probabilistic outcome for a feature or test) and the decision rule is obtained using an optimal binary AND/OR decision tree. An efficient bottom-up tree construction algorithm is developed based on fundamental ideas of Huffman coding. Computational results show that the algorithm outperforms previously proposed heuristic algorithms in reasonable time.

## 3 Unsupervised Learning

Unsupervised learning can be defined as finding patterns in unlabeled data. The goal is finding similarities among instances that belong to same class, where no class information is available. In this group of methods, $N$ instances $(x_1, x_2, \ldots, x_N)$ of a random $p$-vector $\mathbf{X}$ having joint density $Pr(\mathbf{X})$ are given and a decision is to be made based on representation of instances. Unlike the supervised methods that are trying to minimize some external error criterion, in unsupervised methods, calculating the difference between the target and input data instances is not possible. This results in different methods that are not relying on any outside information. It should be noted that unsupervised methods are particularly useful since unlabeled data is usually less costly compared to labeled data. Two widely used applications of unsupervised learning are clustering and dimensionality reduction.

Clustering (or cluster analysis) is defined as finding a convenient and valid organization of the data to establish rules for separating future data into categories [75]. Clustering algorithms can be divided into two major groups: *partitional* and *hierarchical* [74]. In partitional clustering, all clusters are found simultaneously as partitions of the data. In hierarchical clustering, nested clusters are found recursively. This can be done by one of the following two approaches. *Agglomerative methods* initially consider each data point as the head of its own cluster and try to merge the most similar pair of clusters successively to form a cluster hierarchy. On the contrary, *divisive methods* initially assign all data instances in one cluster and divide each cluster into smaller clusters (see [62]).

Another useful application of unsupervised learning is dimensionality reduction. These techniques aim to find a lower dimensional representation of the original data, which captures the content of the original data based on some criterion. A lower

dimensional representation of data is desired mainly because the computational advantage of handling a smaller data set outweighs the loss of information. Frequently used in the preprocessing stage, dimensionality reduction helps mitigate computational limitations of data mining algorithms for large data sets.

## 3.1 Latent Variable Models

In this section, probabilistic methods that utilize *latent variables* are introduced. Latent variable (or hidden variable) is a statistical variable that is not observed directly but can be estimated based on its relation with an observed variable. Latent variable models can be used to perform dimensionality reduction and clustering, the two cornerstones of unsupervised learning. Clustering problems that are addressed in this section are hard clustering problems and fuzzy clustering problems (FCP). In hard clustering problems, each data point needs to be assigned to one and only one cluster. On the other hand, in fuzzy clustering, each data point is a member of all clusters and it has a membership grade for each cluster showing its likelihood of belonging to that cluster.

### 3.1.1 $k$-Means

$k$-means is a hard clustering problem that has been introduced by several researchers across different disciplines, most notably Lloyd [95], Forgy [54], Friedman and Rubin [58], Ball and Hall [6], and MacQueen [96]. Given a data set of $n$ vectors $x_j \in \mathbb{R}^d$, $j = 1, \ldots, n$, the goal is to find a partition $S = \{S_1, S_2, \ldots, S_k\}$ that minimizes the expected loss (see [96]). A random point $z \in \mathbb{R}^d$ with a known distribution $p$ generates a loss proportional to the square of the error resulting from the choice of clusters, formally $\|z - \hat{z}\|^2$, where $\hat{z}$ is the cluster centers or *exemplars*. The goal is to minimize the expected loss over possible choices of clusters, which is formally defined as

$$w^2(S) = \sum_{i=1}^{k} \int_{S_i} \|z - \hat{z}\|^2 dp(z). \tag{20}$$

Given a selection of cluster members from the data, it is easy to see that the cluster mean minimizes the squared error term. Therefore, the problem reduces to selection of cluster members, which can be formulated as the following mixed-integer programming (MIP) problem:

$$[k-\textbf{means}] \quad \min_{\boldsymbol{\mu}, \mathbf{r}} \quad \sum_{i=1}^{k} \sum_{j=1}^{n} r_{ij} \|x_j - \mu_i\|^2 \tag{21a}$$

$$\text{subject to} \quad \sum_{i=1}^{k} r_{ij} = 1 \qquad j = 1, \ldots, n \tag{21b}$$

$$r_{ij} \in \{0,1\} \qquad i = 1,\ldots,k,\ j = 1,\ldots,n. \quad (21c)$$

This problem is proven to be $\mathcal{NP}$-hard [3]. One of the most widely used iterative heuristic approaches is the $k$-means algorithm, also known as Lloyd's algorithm [95], which can be considered as an expectation-maximization algorithm. Initially, $k$ initial exemplars are randomly selected. Next, by repeating the following two steps, the clusters will be iteratively refined. In the first (i.e., *expectation*) step, each instance will be temporarily associated with the cluster whose exemplar is the closest. In the second (i.e., *maximization*) step, based on the temporary cluster assignment, each exemplar is updated as the mean of instances in its cluster. The algorithm terminates when no exemplar can be further updated. The convergence is guaranteed since the objective is finite and an improvement is assured in each step.

It should be noted that the term $k$-means is used to define the problem of minimizing the within-cluster sum of squares (i.e., (21)) as well as Lloyd's expectation-maximization algorithm defined above.

One observation is that $k$-means algorithm is sensitive to the initial set of exemplars especially when $k$ is large. Thus, random initialization of exemplars is acceptable for relatively small values of $k$. There are alternative approaches, especially useful when number of clusters is increased, including, but not limited to, heuristic initialization schemes and parallel implementations with multiple initial exemplars.

Typically, $k$-means problem considers spherical clusters by minimizing Euclidean distance between points and cluster centers, as presented in formulations (20) and (21). Other distances that are used in $k$-means are mahalanobis distance metric [99], Itakura-Saito distance [93], L1 distance [81], and Bregman distance [7].

It has been proven that the worst-case complexity of the $k$-means algorithm is $O(kn^2\Delta^2)$ [4]. Although it is introduced over 50 years ago, $k$-means is still one of the most widely used algorithms for clustering. This algorithm is particularly useful due to its simplicity, ease of implementation, efficiency, and success in practice.

### 3.1.2 Message Passing

In order to overcome difficulties associated with the sensitivity of $k$-means to the initialization process, Frey and Dueck [57] propose a message-passing method where every data instance is a potential exemplar. The problem to be solved is the $\mathcal{NP}$-hard $k$-median problem. For computational tractability, an *affinity propagation* scheme is proposed, where data instances are nodes of a network, real-valued messages are transmitted along edges, and the magnitude of each message is updated based on the current affinity that one data instance has for choosing another data instance as its exemplar. Two types of messages are transmitted: First, *responsibility*, $r(i,j)$, sent from data instance $i$ to candidate exemplar $j$, is a cumulative variable for how appropriate instance $j$ is to be the exemplar

for instance $i$. Second, *availability*, $a(i, j)$, sent from potential exemplar $j$ to data instance $i$, reflects the accumulated evidence for how good it is for instance $i$ to select instance $j$ as its exemplar. Availability includes support from other instances that instance $j$ is an exemplar. Formally, responsibility and availability are defined as

$$r(i, j) \leftarrow s(i, j) - \max_{j' s.t. j' \neq j} \{a(i, j') + s(i, j')\} \tag{22}$$

$$a(i, j) \leftarrow \min \left\{ 0,\ r(j, j) + \sum_{i' s.t. i' \notin \{i, j\}} \max\{0, r(i', j)\} \right\}, \tag{23}$$

where $s(i, j)$ is the similarity between instances $i$ and $j$. The objective is to minimize the total squared error; thus, $s(i, j)$ is defined as $-\|x_i - x_j\|^2$. Since availabilities are zero in the first iteration, $r(i, j)$ is equal to the input similarity between instance $i$ and instance $j$ as its exemplar minus the largest of the similarities between instance $i$ and other potential exemplars. In the subsequent iterations, when an instance is effectively assigned to an exemplar, its updated availabilities will become below zero. This negative availability removes some of the exemplars from candidate exemplars of that data instance. If availability of some other instances becomes negative for that exemplar, the exemplar is out of competition.

When $j = i$, the responsibility $r(j, j)$ equals $s(j, j)$, minus the largest of the similarities between instance $i$ and all other candidate exemplars. If $r(j, j)$ (i.e., self-responsibility) is negative at any iteration, it is more appropriate for instance $j$ to belong to another exemplar rather than being an exemplar itself. This transmission continues until a good set of exemplars and corresponding clusters are obtained.

### 3.1.3 Fuzzy Clustering

Fuzzy clustering is an extension of $k$-means problem, in which instances have a degree of membership for *all* clusters. In this setting, a data instance is not designated to a specific cluster, but it belongs to each cluster to a certain extent, referred to as the *membership grade*. To solve this problem, different nonexact (i.e., soft computing) approaches have been introduced in the literature (see [8, 17, 82, 102]). An exact reformulation-linearization technique (RLT)-based approach has been introduced by Sherali and Desai [123], where the objective is to minimize the total degree-2 fuzzifier weighted squared Euclidean distance. Formally, given a data set of $n$ vectors $a_j \in \mathbb{R}^d$, $j = 1, \ldots, n$, the problem is

$$\textbf{[FCP]} \quad \min_{\mathbf{w}, \mathbf{z}} \quad \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^2 \|a_i - z_j\|^2 \tag{24a}$$

$$\text{subject to} \quad \sum_{j=1}^{c} w_{ij} = 1 \qquad \forall i = 1, \ldots, n \tag{24b}$$

$$w_{ij} \geq 0 \qquad \forall (i, j), \tag{24c}$$

which is essentially a weighted version of (21). Since the objective function (24a) is nonlinear, RLT is used to relax (24) as a linear programming problem and a specialized branch and bound algorithm is employed. To further speed up the process, data reduction is performed on instances by replacing group examples with their centroids [53].

## 3.2 Network-Based Models

In the last few decades, network-based data mining models have received increasing attention for their capability of extracting useful information from large-scale data sets that are widely available in network structure. Boginski [22] presents a wide variety of applications for network-based data mining models.

One use of networks is finding the structural properties of a data set by *degree* (i.e., number of edges emanating from a node) distribution of constructed graphs, which is a representation of large-scale pattern of connections in the graph. This leads to the fact that graphs that are related to completely different data sets might have a similar well-defined power-law structure. Power-law structure states that the probability that a vertex of a graph has degree $k$ is $P(k) \propto k^{-\gamma}$ [22].

Network-based models are widely used in clustering. Finding *cliques* and *independent sets* in a network is synonymous with discovering clusters in that data set. Given a graph $G = (V, E)$, a subgraph whose vertices are pairwise adjacent is called a clique or complete subgraph. On the contrary, an independent set or stable set is a set of vertices none of which are adjacent. Therefore, maximum clique in a graph gives the maximum possible size of a group of similar objects, whereas maximum independent set leads to the largest group of essentially different objects. It should be noted that a clique in graph is an independent set in the complement graph and vice versa. Maximum clique problem and the maximum independent set problem are proven to be $\mathcal{NP}$-hard [61].

These two problems are extended to find the minimum number of distinct cliques for clustering purposes [22]. There also exist relaxations of clique problems such as quasi-clique problems ensuring a lower bound on the total number of edges within the subgraph [1] and $k$-plex problems ensuring a lower bound on the number of adjacent nodes for each selected node within the subgraph [5]. See [125] for a detailed survey on combinatorial optimization techniques for network-based data mining.

### 3.3 Biclustering

Biclustering is simultaneous partitioning of data instances and their features into classes. Instances and features classified together are expected to have a high relevance with each other (i.e., similar data values).

Given an $m \times n$ data matrix $A$, each column represents a data instance and each row represents a feature. Formally, $A = (a_{ij})_{m\times n}$, where $a_{ij}$ is the expression of $i$th feature of $j$th instance. Biclustering is applied by simultaneous partitioning of the instances and features into $k$ classes. Let $S_1, S_2, \ldots, S_k$ denote the classes of the instances (columns) and $F_1, F_2, \ldots, F_k$ denote the classes of features (rows). Biclustering is defined as a collection of pairs of instance and feature subsets $\mathcal{B} = \{(S_1, F_1), (S_2, F_2), \ldots, (S_k, F_k)\}$ such that

$$S_1, S_2, \ldots, S_k \subseteq \{a^j\}_{j=1,\ldots,n},$$
$$\bigcup_{r=1}^{k} S_r = \{a^j\}_{j=1,\ldots,n},$$
$$S_\zeta \bigcap S_\xi = \emptyset \Leftrightarrow \zeta \neq \xi,$$

$$F_1, F_2, \ldots, F_k \subseteq \{a_i\}_{i=1,\ldots,m},$$
$$\bigcup_{r=1}^{k} F_r = \{a_i\}_{i=1,\ldots,m},$$
$$F_\zeta \bigcap F_\xi = \emptyset \Leftrightarrow \zeta \neq \xi,$$

where $\{a^j\}_{j=1,\ldots,n}$ and $\{a_i\}_{i=1,\ldots,m}$ denote the set of columns and rows of the matrix $A$, respectively. The ultimate goal in a biclustering problem is to find a partitioning scheme for which instances from the same class have *similar* values for that class' characteristic features. One of the early algorithms to obtain an appropriate biclustering is [69], which is known as *block clustering*. Given a biclustering $\mathcal{B}$, the variability of the data in the block $(S_r, F_r)$ is used to measure the quality of partitions. A lower variability in the resulting problem is desired. The number of classes should be fixed in order to avoid a trivial, zero variability solution in which each class consists of only one instance. A more sophisticated approach for biclustering is introduced in [40], where the objective is to minimize the mean squared residual. In this setting, the problem is proven to be $\mathcal{NP}$-hard and a greedy algorithm is proposed to find an approximate solution. A simulated annealing algorithm is proposed in [27]. Dhillon [49] proposes another biclustering method for text mining using a bipartite graph. In the graph, the nodes represent features and instances, and each feature $i$ is connected to an instance $j$ with a link $(i, j)$,

which has a weight $a_{ij}$. The total weight of all links connecting features and instances from different classes is used to measure the quality of a biclustering. A lower value corresponds to a better biclustering. A similar method for microarray data is suggested in [83].

Dhillon et al. [50] treat input data as a joint probability distribution between two discrete sets of random variables. The goal of the method is to find disjoint classes for both variables. A Bayesian biclustering technique based on Gibbs sampling can be found in [122]. A detailed survey on biclustering techniques can be found in [31,97].

## 4 Semi-supervised Learning

Semi-supervised learning is a branch of machine learning that utilizes both labeled and unlabeled data to improve generalization performance. Since unlabeled data is relatively easier to collect, these methods are less expensive than supervised learning methods. Moreover, these methods are more accurate than unsupervised learning because they are using information obtained from labeled data as well.

Data used in these methods are comprised a set to be used for training and a set to be used for testing. Let $X = (x_1, \ldots, x_n)$ be a set that consists of $l$ labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$, $y_i = \pm 1$ and of $u$ unlabeled examples $\{x_i\}_{i=l+1}^{n}$. In the literature, $u$ unlabeled examples are referred to as the *working set* and $l$ labeled examples are referred to as the *training set* [14]. Both training set and working set in this definition are used during the training process. Test set is used to test the accuracy of the learning method as usual.

Based on definition of semi-supervised learning, two types of data are available: the similarity distances for the training and working set and class labels of training set. According to the capability of an algorithm to handle unseen data instances (in the test set), semi-supervised methods can be classified as *transductive* or *inductive*. In transductive learning, training, and working sets are processed but test set cannot be used during the algorithm. The early graph-based methods in semi-supervised learning are transductive. On the other hand, inductive learning algorithms can handle test set as well. Semi-supervised learning algorithms can be divided into three main groups based on the data representation considered. These groups are *manifold assumption*, *cluster assumption*, and *manifold-cluster assumption*. In manifold assumption, the data instances lie on a low-dimensional manifold in the input space. These algorithms generally represent data as a graph, instances as vertices, and pairwise similarities between instances as edge weights. In cluster assumption, decision boundaries between classes must lie in low-density regions as the aim of these algorithms is placing instances with high similarities in the same cluster. Manifold-cluster assumption proposed by Mallapragada et al. [98] uses both manifold and cluster assumptions to overcome weaknesses of both approaches.

One of the earliest semi-supervised learning methods proposed in the literature is *self-training*. This method initially constructs a temporary classifier using only

labeled data. Using this classifier, labels are predicted for the instances in the working set and the classifier is progressively updated [150]. Next, some of the most notable methods for semi-supervised learning are introduced.

### 4.1 Semi-supervised Support Vector Machines (S$^3$VM)

The idea of using SVMs to solve semi-supervised learning problems is introduced by Vapnik and Sterin [138]. The combinatorial formulation for semi-supervised support vector machines is as follows (see [37]):

$$[\mathbf{S^3VM}] \quad \min_{\mathbf{w},b,\boldsymbol{\xi},\mathbf{y}\in\mathbb{R}^u} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\left[\sum_{i=1}^{l}\xi_i\right]^p + C^*\left[\sum_{j=l+1}^{l+u}\xi_i\right]^p \tag{25a}$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot x_i + b) + \xi_i \geq 1 \qquad i = 1,\ldots,l+u \tag{25b}$$

$$\xi_i \geq 0 \qquad i = 1,\ldots,l+u \tag{25c}$$

$$\frac{1}{u}\sum_{i=l+1}^{l+u} y_i = 2r - 1 \tag{25d}$$

$$y_i \in \{-1, 1\} \qquad i = l+1,\ldots,l+u. \tag{25e}$$

In this formulation, $y_i$ are given as input for $i = 1,\ldots,l$ (i.e., training set) and are decision variables for $i = l+1,\ldots,l+u$ (i.e., working set). Equation (25d) is called the *balancing constraint* since it enforces the solutions to be balanced by assigning specified percentages of unlabeled data to positive and negative classes [133]. $r$ is the ratio of positive labels to be assigned to the number of instances in the working set.[1] Usually, $r$ is hard to predict; thus, it is assumed to be equal to the ratio of positive labels in the training set. $C$ and $C^*$ represent the penalty weight for labeled (training) and unlabeled (working) data, respectively. To obtain a good generalization performance, ideally $C$ and $C^*$ should be different (see [38]); however, they are usually assumed equal for the sake of simplicity. Some common penalty functions for the objective are linear (i.e., $p = 1$) and quadratic (i.e., $p = 2$) functions.

Bennett and Demiriz [14] introduce a 0–1 variable $d_j$ for each instance $x_j$ in working set to formulate (25) as a MINLP problem. An instance belongs to positive class if $d_j = 1$ and negative class if $d_j = 0$. With this change, S$^3$VM can be formulated as

[1] An exact equality for balancing constraint is likely to lead to infeasible solutions depending on the number of instances and ratios. Therefore, a subtle adjustment is usually necessary to ensure feasibility.

$$[\mathbf{S^3VM-MINLP}] \quad \min_{\mathbf{w},b,\boldsymbol{\eta},\boldsymbol{\xi},\mathbf{z}} \quad C\left[\sum_{i=1}^{l}\eta_i + \sum_{j=l+1}^{l+u}(\xi_j + z_j)\right] + \|\mathbf{w}\| \tag{26a}$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot x_i - b) + \eta_i \geq 1 \qquad i = 1,\ldots,l \tag{26b}$$

$$\mathbf{w}\cdot x_j - b + \xi_j + M(1-d_j) \geq 1 \qquad j = l+1,\ldots,l+u \tag{26c}$$

$$-(\mathbf{w}\cdot x_j - b) + z_j + Md_j \geq 1 \qquad j = l+1,\ldots,l+u \tag{26d}$$

$$\eta_i \geq 0 \qquad i = 1,\ldots,l \tag{26e}$$

$$\xi_j \geq 0 \qquad j = l+1,\ldots,l+u \tag{26f}$$

$$z_j \geq 0 \qquad j = l+1,\ldots,l+u \tag{26g}$$

$$d_j \in \{0,1\} \qquad j = l+1,\ldots,l+u, \tag{26h}$$

where $M$ is a large enough number that ensures $\xi_j = 0$ when $d_j = 0$ and $z_j = 0$ when $d_j = 1$. It should be noted that Bennett and Demiriz [14] assume $C = C^*$, ignore balancing constraint, and use one-norm of $\mathbf{w}$ in the objective function.

A number of methods have been introduced to solve non-convex problem (25) (see [38]). These methods typically utilize continuous relaxations or heuristics except one hard approach solving (25) directly.

### 4.1.1 Branch and Bound

Proposed briefly as a framework in [137], this method is used to find the global optimum for (25). Chapelle et al. [37] implemented a branch and bound algorithm that performs a search over $y_u$'s. This approach is particularly useful for relatively smaller data sets due to its complexity. Based on empirical evidence, the generalization performance of the global optimum solution can be significantly better than nonexact solutions.

The initial solution for the root node of branch and bound tree is the SVM solution for labeled training data. Branching is performed on unlabeled instances in the working set. To achieve optimal solution quickly, unlabeled instance $x^*$ to be labeled next is selected in a way that its label ($y^*$) assignment has a high potential to improve the objective function. To formulate this concept, let $s(L)$ be the SVM objective function trained on labeled set that is formally defined as

$$s(L) = \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{(x_i,y_i)} \max(0, 1 - y_i(\mathbf{w}\cdot x_i + b))^2. \tag{27}$$

The branching is performed on the following unlabeled instance:

$$\arg\max_{x \in U, y \in \pm 1} s(L \cup \{x, y\}). \tag{28}$$

The lower bound for each node will be the objective function value after optimizing a standard SVM on instances that have been labeled so far, i.e., $s(L)$ introduced in (27). The incumbent solution will set the upper bound; thus, exploration strategy in the tree is depth first search. This approach promotes reaching leaves frequently and having tight upper bounds to perform aggressive pruning.

### 4.1.2 Continuous Relaxations

To solve the problem by continuous optimization methods, integer variables $y_i$ for $i = l+1, \ldots, l+u$ should be relaxed. Chapelle et al. [36, 38] introduce a soft computing approach that balances the distance of unlabeled instances from the hyperplane instead of the number of instances as in (25d). The problem is formulated as

$$[\mathbf{S^3VM-R}] \quad \min_{\mathbf{w},b,\boldsymbol{\xi},\mathbf{z}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i^p + C^*\sum_{j=l+1}^{l+u} z_j^p \tag{29a}$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot x_i + b) \geq 1-\xi_i \qquad i = 1,\ldots,l \tag{29b}$$

$$\xi_i \geq 0 \qquad i = 1,\ldots,l \tag{29c}$$

$$|\mathbf{w}\cdot x_j + b| \geq 1 - z_j \qquad j = l+1,\ldots,l+u \tag{29d}$$

$$z_j \geq 0 \qquad j = l+1,\ldots,l+u \tag{29e}$$

$$\frac{1}{u}\sum_{j=l+1}^{l+u} \mathbf{w}^T x_j = 2\widetilde{r} - 1. \tag{29f}$$

Note that (29f) balances the sum of distances for the working set, given $\widetilde{r}$. This constraint is first proposed in [35], and the problem with and without the balancing constraint is solved in [36, 38], respectively.

The problem can be solved via a concave convex-procedure, where the nonconvex objective function is rewritten as the sum of a convex component and a concave component [148]. At each iteration of this algorithm, concave part is estimated by its tangent and the convex part is minimized using traditional algorithms. This type of methods to minimize the concave function is discussed by Fung and Mangasarian [59] and improved in [44, 139] to handle larger data sets.

$\nabla$S$^3$VM method [35] minimizes the objective function by gradient descent. Since the objective function is not smooth, $z_j^p$ in the objective function is replaced by an exponential function $\exp(-s(\mathbf{w}\cdot x_j + b)^p)$, where $s$ is a parameter. Furthermore, an annealing procedure is performed by increasing $C^*$ at each iteration. Chapelle et al. [36] propose a continuation technique to solve the S$^3$VM problem. In this method, a system of nonlinear equations is solved through a simpler system of equations

by smoothing the objective function. Annealing is performed without increasing $C^*$, but instead keeping it fixed, and continuation technique is used to transform the objective function. Chapelle [34] later introduces Newton $S^3$VM that is more efficient compared to $\nabla S^3$VM and Continuation $S^3$VM. This technique utilizes a new loss function for unlabeled instances in the primal problem and an associated Newton method.

Bie and Cristianini [18] propose a convex relaxation using the dual of (25) and solve through semidefinite programming (SDP). The dual formulation also allows kernel trick to be used for nonlinear classification and classification of nonvectorial data. The dual problem can be formulated as

$$\min_{\mathbf{Y}} \max_{\boldsymbol{\alpha}} \quad 2\alpha^T 1 - \alpha^T (K \odot \mathbf{Y}\mathbf{Y}^T)\alpha \tag{30a}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C \qquad i = 1, \ldots, l+u \tag{30b}$$

$$\mathbf{Y}_u \in \{-1, 1\}^u. \tag{30c}$$

In this formulation, $\alpha = (\alpha_1, \ldots, \alpha_{l+u})$ is a vector of dual variables $\alpha_i$, $K$ is the complete kernel matrix, and $\mathbf{Y} = \begin{bmatrix} \begin{pmatrix} \mathbf{Y}_l \\ \mathbf{Y}_u \end{pmatrix} \end{bmatrix}$ is the complete label vector. The optimization problem (30) can be reformulated as follows by introducing the outer product of labels $\Gamma = \mathbf{Y}\mathbf{Y}^T$:

$$\min_{\Gamma} \max_{\boldsymbol{\alpha}} \quad 2\alpha^T 1 - \alpha^T (K \odot \Gamma)\alpha \tag{31a}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C \qquad i = 1, \ldots, l+u \tag{31b}$$

$$\mathbf{Y}_u \in \{-1, 1\}^u. \tag{31c}$$

In (31), $\Gamma = \mathbf{Y}\mathbf{Y}^T = \begin{bmatrix} \begin{pmatrix} \mathbf{Y}_l\mathbf{Y}_l^T & \mathbf{Y}_l\mathbf{Y}_u^T \\ \mathbf{Y}_u\mathbf{Y}_l^T & \mathbf{Y}_u\mathbf{Y}_u^T \end{pmatrix} \end{bmatrix}$, objective function is linear in $\Gamma$, concave in $\alpha$, and constraints are linear. The problem is not convex due to (31c); therefore, a relaxation is proposed as

$$[\mathbf{S^3VM - SDP}] \quad \min_{\Gamma} \max_{\alpha} \quad 2\alpha^T 1 - \alpha^T (K \odot \Gamma)\alpha \tag{32a}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C \qquad i = 1, \ldots, l+u \tag{32b}$$

$$\text{diag}(\Gamma) = 1 \tag{32c}$$

$$\Gamma \succeq 0, \tag{32d}$$

where $\Gamma = \begin{pmatrix} \mathbf{Y}_l\mathbf{Y}_l^T & \Gamma_{lu} \\ \Gamma_{ul} & \Gamma_{uu} \end{pmatrix}$. In (32), $\mathbf{Y}_u$'s lie in the interval $[-1, 1]$. Furthermore, rank of $\Gamma$ is not necessarily 1. Although the problem is convex, this method is not widely used due to its high time complexity $O((1+u^2)^2(l+u)^{2.5})$ [38].

### 4.1.3 Heuristics

Introduced by Thorsten [133], $S^3VM^{light}$ is an $S^3VM$ implementation within the $SVM^{light}$ software package. It is based on local combinatorial search guided by a label-switching procedure. In this method, the labels of test instances are switched in a way that improves the objective function. In an outer loop, the value of $C^*$ is gradually increased. This temporary adjustment of $C^*$ helps algorithm avoid suboptimal local minima.

Deterministic annealing (DA) is a metaheuristic that has been used to solve hard combinatorial or non-convex problems. Sindhwani and Keerthi [127] use DA for $S^3VMs$ by relaxing discrete label variables $y_u$ to real-valued variables $p_u = (p_{l+1}, \ldots, p_{l+u})$, where $p_i$ is interpreted as the probability that $y_i = 1$. Based on these new variables, an easier reformulation is solved.

## 4.2 Expectation-Maximization (EM) Method

The expectation-maximization (EM) algorithm is first introduced by Dempster et al. [48] as an iterative procedure for estimating parameter values that maximize the likelihood function, when there is missing data. This method has two steps: *expectation* (E-step) and *maximization* (M-step). E-step calculates the expected values of the sufficient statistics given the current parameter estimates. M-step sets parameters to their maximum likelihood estimates given the estimated values of the sufficient statistics. Starting with a randomly generated set of parameter estimates, these steps are repeated either for a predefined number of steps or more commonly until the difference between parameter estimates in two consecutive iterations is negligible. In this setting, labels for working set are considered as missing data and EM method attempts to label these instances [100].

## 4.3 Graph-Based Methods

Graph-based semi-supervised methods define an empirical graph $G = (V, E)$. In this graph, nodes $V = 1, \ldots, n$ are instances given in training and working sets, and edges $E$ represent the similarity between these instances. A weight matrix $W = [w_{ij}]$ is also defined in a way that $w_{ij} \neq 0$ if there is an edge between $i$ and $j$. Weight matrix can be introduced in a number of different ways. For example, $W$ can be a $k$-nearest neighbor matrix, i.e., $w_{ij} = 1$ if $i$ is among the $k$-nearest neighbors of $j$. The way the weight matrix is defined affects the performance of graph-based semi-supervised learning [41].

These methods rely on the geometry of data induced by labeled and unlabeled instances and usually assume a smooth distribution of labels over the graph. Graph-based methods are nonparametric, discriminative, and transductive in nature, yet they can be modified to be inductive. Goal of a graph-based method can be *label propagation* on a similarity graph or minimization of a *quadratic cost criterion*.

In label propagation, known labels are used to spread information through the graph in order to label unlabeled nodes [151]. Szummer and Jaakkola [132] propose a Markov random walk algorithm based on label propagation on similarity graph. In this method, labeling is performed based on how easy it gets to move from one node to the other via a random walk.

In quadratic cost criterion-based methods, rapid changes of estimated labels (i.e., $\hat{Y}$) between close instances are penalized. The penalty term is defined as follows:

$$\begin{aligned}\frac{1}{2}\sum_{i,j=1}^{n} W_{i,j}(\hat{y}_i - \hat{y}_j)^2 &= \tfrac{1}{2}\left(2\sum_{i=1}^{n}\hat{y}_i^2\sum_{j=1}^{n}W_{ij} - 2\sum_{i,j=1}^{n}W_{ij}\hat{y}_i\hat{y}_j\right)\\ &= \hat{Y}^T(D-W)\hat{Y}\\ &= \hat{Y}^T L\hat{Y}. \end{aligned} \tag{33}$$

In (33), $L = D - W$ is an un-normalized graph laplacian. In [153], the penalty term is minimized over $\hat{Y}_u$. Other methods considering quadratic cost criterion can be found in [9,47,79]. Bengio et al. [13] show that minimizing quadratic cost criterion is equivalent to label propagation.

### 4.4 Co-training

In co-training, two learners are iteratively combining their outputs to increase size of the training set, which is reused for training and generating more labeled data automatically. Co-training is proposed by Blum and Mitchell [21] for semi-supervised learning. This technique assumes features are separable into two sets that are conditionally independent given the classes. In first step, two classifiers are trained with the labeled data using subfeature sets. Then, each classifier is used to designate labels to unlabeled instances. Each classifier is retrained by the most confident predictions given by the other classifier. This process is repeated until all instances in the working set are labeled [149].

## 5 Ranking

Ranking is one of the fastest growing areas in machine learning. In ranking, a group of alternatives are ranked based on aggregate scores assigned by voters. Some of the most popular applications are internet databases, search engines, and e-commerce sites. A number of algorithms have been proposed for ranking in the literature [28,52,56] with different limitations.

Recently, Jiang et al. [76] propose a model called HodgeRank using graph Helmholtzian. Alternatives to be ranked by voters are vertices of a graph, where preferences are quantified and aggregated into an edge flow. Hodge theory yields an orthogonal decomposition of the edge flow known as the Hodge or

Helmholtz decomposition and analyzes pairwise rankings represented as edge flows. HodgeRank uses combinatorial Hodge theory to reduce rank aggregation to a linear least squares regression and avoids usual $\mathcal{NP}$-hard combinatorial optimization problems. HodgeRank outperforms classical ranking methods due to its ability to handle incomplete and imbalanced data as well as large complex network data. Furthermore, although this method is designed for cardinal data (i.e., each alternative is given a score), it can also give insights about ordinal data sets (i.e., a set of alternatives is compared and ranked by each voter).

## 6 Feature Selection

Feature selection has been studied since the 1970s in statistical pattern recognition, machine learning, and data mining [94]. Feature selection is applied in a variety of fields such as text categorization, gene expression array analysis, combinatorial chemistry, and bioinformatics [66, 115] and is one of the key techniques in data preprocessing for data mining [20]. Feature selection is the process of removing irrelevant (features that do not affect the underlying target concept) and redundant features (features that do not add any information to the target concept) [46, 80]. Feature selection's role is more significant in real-world problems in which the data set is large (i.e., larger than the desired learning tool can handle). The idea with feature selection is to decrease running time of the learning process without decreasing the accuracy of the result significantly. It is needed because of generalization performance, running time requirements, constraints, and interpretational issues imposed by the problem itself. Further benefits of feature selection include facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, and defying the curse of dimensionality to improve prediction performance [66]. Problem of selecting the optimal subset of features is proven to be an $\mathcal{NP}$-complete problem [63].

Based on availability of class information, feature selection methods can be divided into three groups: supervised feature selection (feature selection for classification), unsupervised feature selection (feature selection for clustering), and semi-supervised feature selection. Feature selection methods can also be categorized based on role of the mining algorithm: the filter model, the wrapper model, and the hybrid model. *Filter models* start with an empty set of features and add features until a combination consistent with the training data is found. Mining method is applied next based on the selected features. On the other hand, *wrapper models* need a predefined mining method to use as evaluation criterion. Generally, wrapper methods are more accurate than filter methods due to the interaction between a mining algorithm and its training data [84, 90]. Wrapper methods are also computationally more expensive as they run the mining algorithm repeatedly during the evaluation process.

Feature selection in classification is defined as methods that select minimal sized subset of features based on two criteria. First, classification accuracy should

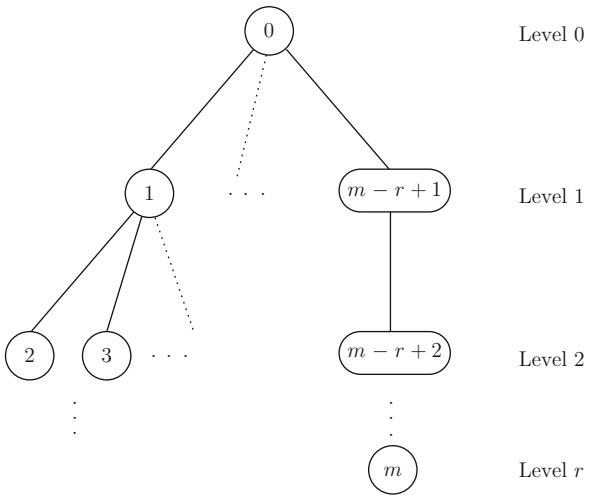


**Fig. 3** Illustration of a branch and bound algorithm for feature selection

not notably decrease. Second, the resulting class distribution based on selected features should be as close as possible to the original class distribution based on all features. There are a number of studies in the literature (e.g., [51, 126]) that explore stages of feature selection methods. Dash and Liu [46] breaks down a typical feature selection method into four basic steps: generation, evaluation, stopping criterion, and validation. Generation procedure is a search approach that generates subsets for evaluation. When number of features is $N$, order of search space is $O(2^N)$. The search approach can be complete, heuristic, or random. Using complete methods guarantees optimality of the feature subset because they do backtracking. Backtracking is done by using techniques such as branch and bound, best first search, and beam search. Stopping criterion can be based on generation (predefined number of features/iterations) or evaluation procedure (optimal subset is obtained or addition/deletion of any feature does not result in a better subset).

Basic branch and bound algorithm omits $r$ out of $m$ features by producing $r$ levels as shown in Fig. 3. At each level, one of the features is discarded. The level number indicates the number of features that have been omitted at that level. The root of the tree (at level 0) refers to the set of all $m$ features. The leaves at the bottom of the tree correspond to all possible subsets of size $m - r$ features. The paths in this tree denote all combinations of $r$-eliminated features. Numbers in the nodes correspond to omitted features. The problem is selecting the best path through the tree that yields the best criterion function result. Ideally, this path should be achieved with the fewest number of calculations.

Branch and bound is first used for solving feature selection problem by Narendra and Fukunaga [105]. This algorithm assumes *monotonicity* of criterion function to find optimal feature subset without evaluating all possible feature subsets. Monotonicity property states that a subset of features is not better than any larger set

that contains the subset. A criterion function that satisfies the monotonicity property helps to cut off some subtrees and decreases the search area in the branch and bound algorithm. Generally, all branch and bound algorithms used for feature selection fall into two groups based on monotonicity assumption. Hamamoto et al. [68] show that criterion function is not monotonic, branch and bound still can result in a good recognition rate. A more efficient branch and bound algorithm with same assumption is introduced by Yu and Yuan [147]. This method can handle high-dimensional data by dynamically searching for the optimal solution on a minimum solution tree which is a subtree of the traditional solution tree. Another high-dimensional branch and bound algorithm (HDBB) that assumes monotonicity is introduced in [32]. To improve the speed of branch and bound, right-left search strategy is employed in addition to top-down strategy and backtracking in [39]. Frank et al. [55] propose a branch and bound algorithm using Bhattacharyya distance which is monotonic and additive. This method finds a feature subset of a given size with the lowest Bayesian classification error. A branch and bound method that can give a large initial bound using a floating search method is introduced in [104]. This model can order the tree nodes by the significance of features and has a jump search strategy to avoid redundant criterion function calculations. There are a number of methods that utilize monotonicity assumption (see [85, 128, 140, 146]). Recently, Ris et al. [114] introduce a branch and bound algorithm, which bases the representation and exploration of the search space on new lattice properties and the criterion function is not monotonic.

Another widely used approach in feature selection is the use of SVMs. Combined with filter methods, SVMs can be used after features are selected. SVMs can also be used in wrapper strategies, where one-norm SVM is used for automatic feature selection [23, 129, 152]. It is shown that the one-norm SVM has advantages over the two-norm SVM when there are features that present noise [152]. Having more spare solutions implies $\mathbf{w}$ has more zero components; thus, less features could be chosen. An extension of one-norm SVM is used in [154]. This method generates groups among features by clustering and uses $F_\infty$-norm SVM. Shi et al. [124] introduce a feature selection strategy using $l_p$-norm support vector classification ($l_p$-SVC) and $l_p$-norm proximal support vector machine ($l_p$-PSVM) where $0 < p < 1$. Guyon et al. [67] have proposed a recursive feature elimination (RFE) method. Adaptive scaling methods can be used for feature selection in SVMs [65]. For scaling, a linear transformation is done within the input space.

## 7 Conclusion

This review presents the numerous applications in data mining, which have benefited from the theory of combinatorial optimization. Combinatorial optimization improves quality and robustness of numerous applications and will certainly continue to support the constantly growing field of data mining.

## Cross-References

▸Combinatorial Optimization Techniques for Network-Based Data Mining

## Recommended Reading

1. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN 2002: Theoretical Informatics* (Springer, Berlin/New York, 2002), pp. 598–612
2. S. Alexe, E. Blackstone, P. Hammer, H. Ishwaran, M. Lauer, C. Snader, Coronary risk prediction by logical analysis of data. Ann. Oper. Res. **119**, 15–42 (2003)
3. D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering. Mach. Learn. **75**, 245–248 (2009)
4. D. Arthur, S. Vassilvitskii, How slow is the k-means method? in *Proceedings of the 22nd Annual Symposium on Computational Geometry* (ACM, New York, 2006), pp. 144–153
5. B. Balasundaram, S. Butenko, I.V. Hicks, Clique relaxations in social network analysis: the maximum k-plex problem. Oper. Res. **59**, 133–142 (2011)
6. G.H. Ball, D.J. Hall, ISODATA, a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, CA, 1965
7. A. Banerjee, S. Merugu, I.S. Dhillon, J. Ghosh, Clustering with Bregman divergences. J. Mach. Learn. Res. **6**, 1705–1749 (2005)
8. A. Baraldi, P. Blonda, A survey of fuzzy clustering algorithms for pattern recognition – part II. IEEE Trans. Syst. Man Cybern. B **29**(6), 786–801 (1999)
9. M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs. Learn. Theory **3120**, 624–638 (2004)
10. A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, Z. Yakhini, Tissue classification with gene expression profiles, in *Proceedings of the 4th Annual International Conference on Computational Biology (RECOMB)*, Tokyo, 2000, pp. 54–64
11. A. Ben-Dor, N. Friedman, Z. Yakhini, Class discovery in gene expression data, in *Proceedings of the 5th Annual International Conference on Computational Biology (RECOMB)*, New York, NY, USA (ACM, 2001), pp. 31–38
12. A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, Discovering local structure in gene expression data: the order-preserving submatrix problem. J. Comput. Biol. **10**(3–4), 373–384 (2003)
13. Y. Bengio, O. Delalleau, N. Le Roux, Label propagation and quadratic criterion, in *Semi Supervised Learning* (MIT, Cambridge, 2006)
14. K.P. Bennett, A. Demiriz, Semi-supervised support vector machines. Adv. Neural Inf. Process. Syst. **11**, 368–374 (1999)
15. C. Bergeron, F. Cheriet, J. Ronsky, R. Zernicke, H. Labelle, Prediction of anterior scoliotic spinal curve from trunk surface using support vector regression. Eng. Appl. Artif. Intell. **18**(8), 973–983 (2005)
16. D. Bertsimas, R. Shioda, Classification and regression via integer optimization. Oper. Res. **55**(2), 252–271 (2007)
17. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms* (Kluwer Academic, Norwell, 1981)
18. T.D. Bie, N. Cristianini, Semi-supervised learning using semi-definite programming, in *Semi-Supervised Learning* (MIT, Cambridge, 2006), pp. 119–135
19. C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer, New York, 2006)
20. A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning. Artif. Intell. **97**(1–2), 245–271 (1997)
21. A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in *Proceedings of the 11th Annual Conference on Computational Learning Theory* (ACM, New York, 1998), pp. 92–100

22. V. Boginski, Network-based data mining: operations research techniques and applications, in *Encyclopedia of Operations Research and Management Science* (Wiley, Hoboken, 2010) pp. 3498–3508
23. P.S. Bradley, O.L. Mangasarian, Feature selection via concave minimization and support vector machines, in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, Madison, 1998, pp. 82–90
24. P.S. Bradley, U.M. Fayyad, O.L. Mangasarian, Mathematical programming for data mining: formulations and challenges. INFORMS J. Comput. **11**, 217–238 (1999)
25. J.P. Brooks, Support vector machines with the ramp loss and the hard margin loss. Oper. Res. **59**(2), 467–479 (2011)
26. M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugne, T. Furey, M. Ares, D. Haussler, Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc. Natl. Acad. Sci. **97**(1), 262–267 (2000)
27. K. Bryan, Biclustering of expression data using simulated annealing, in *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS)* Washington, DC, USA, 2005, pp. 383–388
28. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, 2005, pp. 89–96
29. S. Busygin, O.A. Prokopyev, P.M. Pardalos, Feature selection for consistent biclustering. J. Comb. Optim. **10**, 7–21 (2005)
30. S. Busygin, N. Boyko, P.M. Pardalos, M. Bewernitz, G. Ghacibeh, Biclustering EEG data from epileptic patients treated with vagus nerve stimulation, in *Data Mining, Systems Analysis and Optimization in Biomedicine*, vol. 953, ed. by O. Seref, O.E. Kundakcioglu, P.M. Pardalos (American Institute of Physics, Melville, 2007), pp. 220–231
31. S. Busygin, O. Prokopyev, P.M. Pardalos, Biclustering in data mining. Comput. Oper. Res. **35**(9), 2964–2987 (2008)
32. D. Casasent, X.W. Chen, Waveband selection for hyperspectral data: optimal feature selection, in *Proceedings of SPIE*, vol. 5106, Orlando, FL, 2003, pp. 259–270
33. W. Chaovalitwongse, Novel quadratic programming approach for time series clustering with biomedical application. J. Comb. Optim. **15**, 225–241 (2008)
34. O. Chapelle, Training a support vector machine in the primal. Neural Comput. **19**, 1155–1178 (2007)
35. O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in *Proceeding of International Conference on Artificial Intelligence and Statistics (AISTAT)*, Barbados, 2005, pp. 57–64
36. O. Chapelle, M. Chi, A. Zien, A continuation method for semi-supervised SVMs, in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, New York, NY, USA (ACM, 2006), pp. 185–192
37. O. Chapelle, V. Sindhwani, S.S. Keerthi, Branch and bound for semi-supervised support vector machines. Adv. Neural Inform. Process. Syst. **19**, 217–224 (2007)
38. O. Chapelle, V. Sindhwani, S.S. Keerthi, Optimization techniques for semi-supervised support vector machines. J. Mach. Learn. Res. **9**, 203–233 (2008)
39. X. Chen, An improved branch and bound algorithm for feature selection. Pattern Recognit. Lett. **24**(12), 1925–1933 (2003)
40. Y. Cheng, G.M. Church, Biclustering of expression data, in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology* (AAAI, Menlo Park, 2000) pp. 93–103
41. H. Cheng, Z. Liu, J. Yang, Sparsity induced similarity measure for label propagation, in *Proceedings of 12nd IEEE International Conference on Computer Vision*, Kyoto, Japan, 2010, pp. 317–324
42. K.Y. Choy, C.W. Chan, Modeling of river discharges and rainfall using radial basis function networks based on support vector regression. Int. J. Syst. Sci. **34**(14–15), 763–773 (2003)

43. C. Cifarelli, G. Patrizi, Solving large protein folding problem by a linear complementarity algorithm with 0–1 variables. Optim. Methods Softw. **22**(1), 25–49 (2007)
44. R. Collobert, F. Sinz, J. Weston, L. Bottou, T. Joachims, Large scale transductive SVMs. J. Mach. Learn. Res. **7**, 2006 (2006)
45. N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press, Cambridge, 2000)
46. M. Dash, H. Liu, Feature selection for classification. Intell. Data Anal. **1**(3), 131–156 (1997)
47. O. Delalleau, Y. Bengio, N. Le Roux, Efficient non-parametric function induction in semi-supervised learning, in *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, Barbados, 2005
48. A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. B **39**(1), 1–38 (1977)
49. I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA (ACM, 2001), pp. 269–274
50. I.S. Dhillon, S. Mallela, D.S. Modha, Information-theoretic co-clustering, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA (ACM, 2003), pp. 89–98
51. J. Doak, An evaluation of feature selection methods and their application to computer security. Technical report, University of California, 1992
52. C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in *Proceedings of the 10th International Conference on World Wide Web*, New York, NY, USA (ACM, 2001), pp. 613–622
53. S. Eschrich, J. Ke, L.O. Hall, D.B. Goldgof, Fast accurate fuzzy clustering through data reduction. IEEE Trans. Fuzzy Syst. **11**(2), 262–270 (2003)
54. E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classifications. Biometrics **21**(3), 768 (1965)
55. A. Frank, D. Geiger, Z. Yakhini, A distance-based branch and bound feature selection algorithm, in *Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, Acapulco, 2003, pp. 241–248
56. Y. Freund, R. Iyer, R.E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res. **4**, 933–969 (2003)
57. B.J. Frey, D. Dueck, Clustering by passing messages between data points. Sci. **315**(5814), 972–976 (2007)
58. H.P. Friedman, J. Rubin, On some invariant criteria for grouping data. J. Am. Stat. Assoc. **62**(320), 1159–1178 (1967)
59. G. Fung, O.L. Mangasarian, Semi-supervised support vector machines for unlabeled data classification. Optim. Methods Softw. **15**, 29–44 (2001)
60. G.N. Garcia, T. Ebrahimi, J.M. Vesin, Joint time-frequency-space classification of EEG in a brain-computer interface application. J. Appl. Signal Process **7**, 713–729 (2003)
61. M.R. Garey, D.S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York, 1979)
62. Z. Ghahramani, Unsupervised learning, in *Advanced Lectures on Machine Learning* (Springer, Berlin/New York, 2003), pp. 72–112
63. I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains. Pattern Recognit. **43**(1), 5–13 (2010)
64. T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science **286**(5439), 531–537 (1999)
65. Y. Grandvalet, S. Canu, Adaptive scaling for feature selection in SVMs, in *NIPS*, Vancouver, 2002, pp. 553–560
66. I. Guyon, A. Elisseeff, An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)

67. I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines. Mach. Learn. **46**, 389–422 (2002)
68. Y. Hamamoto, S. Uchimura, Y. Matsuura, T. Kanaoka, S. Tomita, Evaluation of the branch and bound algorithm for feature selection. Pattern Recognit. Lett. **11**(7), 453–456 (1990)
69. J.A. Hartigan, Direct clustering of a data matrix. J. Am. Stat. Assoc. **67**(337), 123–129 (1972)
70. W.C. Hong, P.F. Pai, Potential assessment of the support vector regression technique in rainfall forecasting. Water Res. Manage. **21**(2), 495–513 (2007)
71. C.W. Hsu, C.C. Chang, C.J. Lin, A practical guide to support vector classification (2004), http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
72. Z. Huang, H. Chen, C.J. Hsu, W.H. Chenb, S. Wuc, Credit rating analysis with support vector machines and neural networks: a market comparative study. Decis. Support Syst. **37**, 543–558 (2004)
73. K. Hyunsoo, Z.X. Jeff, M.C. Herbert, P. Haesun, A three-stage framework for gene expression data analysis by L1-norm support vector regression. Int. J. Bioinformatics Res. Appl. **1**(1), 51–62 (2005)
74. A.K. Jain, Data clustering: 50 years beyond k-means. Pattern Recognit. Lett. **31**(8), 651–666 (2010)
75. A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Upper Saddle River, 1988)
76. X. Jiang, L.H. Lim, Y. Yao, Y. Ye, Statistical ranking and combinatorial hodge theory. Mathematical Programming **127**, 1–42 (2010)
77. T. Joachims, Text categorization with support vector machines: learning with many relevant features, in *Proceedings of the European Conference on Machine Learning*, Berlin, ed. by C. Nédellec, C. Rouveirol (Springer, 1998), pp. 137–142
78. T. Joachims, Making large–scale SVM learning practical, in *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA, ed. by B. Schölkopf, C.J.C. Burges, A.J. Smola (MIT, 1999), pp. 169–184
79. T. Joachims, Transductive learning via spectral graph partitioning, in *Proceedings of 20th International Conference on Machine Learning (ICML)*, Washington, DC, USA, vol. 20, 2003, pp. 290–297
80. G.H. John, R. Kohavi, K. Pfleger, Irrelevant features and the subset selection problem, in *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, vol. 129, 1994, pp. 121–129
81. H. Kashima, J. Hu, B. Ray, M. Singh, K-means clustering of proportional data using L1 distance, in *Proceedings of 19th International Conference on Pattern Recognition (ICPR)*, Tampa, FL, 2009, pp. 1–4
82. F. Klawonn, A. Keller, Fuzzy clustering based on modified distance measures, in *IDA '99 Proceedings of the Third International Symposium on Advances in Intelligent Data Analysis* (Springer, Berlin, 1999), pp. 291–302
83. Y. Kluger, R. Basri, J.T. Chang, M. Gerstein, Spectral biclustering of microarray data: coclustering genes and conditions. Genome Res. **13**(4), 703–716 (2003)
84. R. Kohavi, G.H. John, Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)
85. M. Kudo, J. Sklansky, Comparison of algorithms that select features for pattern classifiers. Pattern Recognit. **33**(1), 25–41 (2000)
86. O.E. Kundakcioglu, P.M. Pardalos, The complexity of feature selection for consistent biclustering, in *Clustering Challenges in Biological Networks* (World Scientific, Hackensack, 2009), pp. 257–266
87. O.E. Kundakcioglu, T. Ünlüyurt, Bottom-up construction of minimum-cost AND/OR trees for sequential fault diagnosis. IEEE Trans. Syst. Man Cybern. A **37**(5), 621–629 (2007)
88. O.E. Kundakcioglu, O. Seref, P.M. Pardalos, Multiple instance learning via margin maximization. Appl. Numer. Math. **60**(4), 358–369 (2010)

89. T.N. Lal, M. Schroeder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, B. Schölkopf, Support vector channel selection in BCI. IEEE Trans. Biomed. Eng. **51**(6), 1003–1010 (2004)
90. P. Langley, Selection of relevant features in machine learning, in *Proceedings of the AAAI Fall Symposium on Relevance* (AAAI, 1994), New Orleans, LA, pp. 140–144
91. F. Lauer, G. Bloch, Incorporating prior knowledge in support vector regression. Mach. Learn. **70**, 89–118 (2008)
92. S. Lee, A. Verri (eds.), *Pattern Recognition with Support Vector Machines*, Niagara Falls, Canada (Springer, New York/Berlin, 2002)
93. Y. Linde, A. Buzo, R. Gray, An algorithm for vector quantizer design. IEEE Trans. Commun. **28**(1), 84–95 (1980)
94. H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowl. Data Eng. **17**(4), 491–502 (2005)
95. S. Lloyd, Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**, 129–137 (1982). Original paper was published as a technical note in 1957, Bell Labs
96. J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in *Fifth Symposium on Math, Statistics and Probability* (University of California Press, Berkeley, 1967), pp. 281–297
97. S. Madeira, A. Oliveira, Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Trans. Comput. Biol. Bioinformatics **1**, 24–45 (2004)
98. P.K. Mallapragada, R. Jin, A.K. Jain, Y. Liu, SemiBoost: boosting for semi-supervised Learning. IEEE Trans. Pattern Anal. Mach. Intell. **31**(11), 2000–2014 (2009)
99. J. Mao, A.K. Jain, A self-organizing network for hyperellipsoidal clustering (HEC). IEEE Trans. Neural Netw. **7**(1), 16–29 (2002)
100. G.J. McLachlan, T. Krishnan, *The EM algorithm and extensions* Wiley-Interscience, Hoboken, Newjersy (LibreDigital, 2008)
101. Merriam-Webster, Dictionary and Thesaurus – Merriam-Webster Online (2011), http://www.merriam-webster.com/dictionary/data_mining
102. B.G. Mirkin, *Mathematical Classification and Clustering*, Kluwer Academic Publishers, Dordrecht, Netherland, (Springer, 1996)
103. A. Nahapetyan, S. Busygin, P.M. Pardalos, An improved heuristic for consistent biclustering problems, in *Mathematical Modelling of Biosystems* (Springer, Berlin, 2008), pp. 185–198
104. S. Nakariyakul, D.P. Casasent, Adaptive branch and bound algorithm for selecting optimal features. Pattern Recognit. Lett. **28**(12), 1415–1427 (2007)
105. P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection. IEEE Transact. Comput. **100**(9), 917–922 (1977)
106. W.S. Noble, Support vector machine applications in computational biology, in *Kernel Methods in Computational Biology* (MIT, Cambridge MA, 2004), New York, NY, pp. 71–92
107. R.F.E. Osuna, F. Girosi, An improved training algorithm for support vector machines, in *IEEE Workshop on Neural Networks for Signal Processing*, New York, NY, 1997, pp. 276–285
108. P.F. Pai, W.C. Hong, A recurrent support vector regression model in rainfall forecasting. Hydrol. Process. **21**(6), 819–827 (2007)
109. P.M. Pardalos, E. Romeijn (eds.), *Handbook of Optimization in Medicine* (Springer, Newyork/London, 2009)
110. J. Platt, Fast training of SVMs using sequential minimal optimization, in *Advances in Kernel Methods: Support Vector Learning* (MIT, Cambridge MA, 1999), pp. 185–208
111. M.H. Poursaeidi and O.E. Kundakcioglu, Robust support vector machines for multiple instanceclassification, Annals of Operations Research, published online. doi:10.1007/s10479-012- 1241-z
M.H. Poursaeidi, O.E. Kundakcioglu, Robust support vector machines for multiple instance classification (2011, under revision)

112. G. Pyrgiotakis, O.E. Kundakcioglu, K. Finton, P.M. Pardalos, K. Powers, B.M. Moudgil, Cell death discrimination with Raman spectroscopy and support vector machines. Ann. Biomed. Eng. **37**(7), 1464–1473 (2009)
113. G. Pyrgiotakis, O.E. Kundakcioglu, P.M. Pardalos, B.M. Moudgil, Raman spectroscopy and support vector machines for quick toxicological evaluation of titania nanoparticles. J. Raman Spectrosc. (2011, accepted). doi:10.1002/jrs.2839
114. M. Ris, J. Barrera, D.C. Martins Jr., U-curve: a branch-and-bound optimization algorithm for u-shaped cost functions on boolean lattices applied to the feature selection problem. Pattern Recognit. **43**(3), 557–568 (2010)
115. Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics. Bioinformatics **23**(19), 2507 (2007)
116. N.A. Sakhanenko, G.F. Luger, Shock physics data reconstruction using support vector regression. Int. J. Mod. Phys. **17**(9), 1313–1325 (2006)
117. B. Schölkopf, A.J. Smola, *Learning with Kernels* (MIT, Cambridge MA, 2002)
118. O. Seref, O.E. Kundakcioglu, P.M. Pardalos, Selective linear and nonlinear classification, in *CRM Proceedings and Lecture Notes*, vol. 45, ed. by P.M. Pardalos, P. Hansen (American Mathematical Society, Providence, 2008), pp. 211–234
119. O. Seref, O.E. Kundakcioglu, O.A. Prokopyev, P.M. Pardalos, Selective support vector machines. J. Comb. Optim. **17**(1), 3–20 (2009)
120. S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. B **127**, 3–30 (2011)
121. J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis* (Cambridge University Press, Cambridge, 2004)
122. Q. Sheng, Y. Moreau, B. DeMoor, Biclustering microarray data by Gibbs sampling. Bioinformatics **19**, 196–205 (2003)
123. H.D. Sherali, J. Desai, A global optimization RLT-based approach for solving the fuzzy clustering problem. J. Glob. Optim. **33**(4), 597–615 (2005)
124. Y. Shi, Y. Tian, G. Kou, Y. Peng, J. Li, *Optimization Based Data Mining: Theory and Applications* (Springer, New York, 2011)
125. O. Shirokikh, V. Stozhkov, V. Boginski, Combinatorial optimization techniques for network-based data mining, in *Handbook of Combinatorial Optimization, 2nd Edition*, (Springer, 2013)
126. W. Siedlecki, J. Sklansky, On automatic feature selection. Intern. J. Pattern Recognit. Artif. Intell. **2**(2), 197–220 (1988)
127. V. Sindhwani, S.S. Keerthi, Large scale semi-supervised linear SVMs, in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (ACM, New York, 2006), pp. 477–484
128. P. Somol, P. Pudil, J. Kittler, Fast branch & bound algorithms for optimal feature selection. IEEE Trans. Pattern Anal. Mach. Intell. **2**6(7), 900–912 (2004)
129. M. Song, C.M. Breneman, J. Bi, N. Sukumar, K.P. Bennett, S. Cramer, N. Tugcu, Prediction of protein retention times in anion-exchange chromatography systems using support vector regression. J. Chem. Inf. Comput. Sci. **42**(6), 1347–1357 (2002)
130. I. Steinwart, Support vector machines are universally consistent. J. Complex. **18**, 768–791 (2002)
131. Y.F. Sun, Y.C. Liang, C.G. Wu, X.W. Yang, H.P. Lee, W.Z. Lin, Estimate of error bounds in the improved support vector regression. Prog. Nat. Sci. **14**(4), 362–364 (2004)
132. M. Szummer, T. Jaakkola, Partially labeled classification with Markov random walks. Adv. Neural Inf. Process. Syst. **2**, 945–952 (2002)
133. J. Thorsten, Transductive inference for text classification using support vector machines, in *Proceedings of 16th International Conference on Machine Learning* (Morgan Kaufmann, San Francisco, 1999), pp. 200–209
134. T.B. Trafalis, H. Ince, Support vector machine for regression and applications to financial forecasting, in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, Como, 2002

135. A.C. Trapp, O.A. Prokopyev, Solving the order-preserving submatrix problem via integer programming. INFORMS J. Comput. **22**(3), 387–400 (2010)
136. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer, New York, 1995)
137. V. Vapnik, A. Chervonenkis, *Theory of Pattern Recognition* (Naula/Moscow, Russia, 1974)
138. V. Vapnik, A. Sterin, On structural risk minimization or overall risk in a problem of pattern recognition, in *Automation and Remote Control*, vol. 10, 1977, pp. 1495–1503
139. J. Wang, On transductive support vector machines, in *Prediction and Discovery* (American Mathematical Society, Providence, Snowbird, Utah, 2007)
140. Z. Wang, J. Yang, G. Li, An improved branch & bound algorithm in feature selection, in *Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, Chongqing, 2003, pp. 549–556
141. J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMs, in *Proceeding of NIPS*, Denver, 2000, pp. 668–674
142. Z.L. Wu, C.H. Li, J.K.Y. Ng, K.R.P.H. Leung, Location estimation via support vector regression. IEEE Trans. Mob. Comput. **6**(3), 311–321 (2007)
143. X.S. Xie, W.T. Liu, B.Y. Tang, Space based estimation of moisture transport in marine atmosphere using support vector regression. Remote Sens. Environ. **112**(4), 1846–1855 (2008)
144. E.P. Xing, R.M. Karp, CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. Bioinformatics Discov. Note **17**, 306–315 (2001)
145. K. Yamamoto, F. Asano, T. Yamada, N. Kitawaki, Detection of overlapping speech in meetings using support vector machines and support vector regression. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E89–A**(8), 2158–2165 (2006)
146. S. Yang, P. Shi, Bidirectional automated branch and bound algorithm for feature selection. J. Shanghai Univ. (English Edition) **9**(3), 244–248 (2005)
147. B. Yu, B. Yuan, A more efficient branch and bound algorithm for feature selection. Pattern Recognit. **26**(6), 883–889 (1993)
148. A.L. Yuille, A. Rangarajan, The concave-convex procedure. Neural Comput. **15**(4), 915–936 (2003)
149. X. Zhu, Semi-supervised learning with graphs. PhD thesis, Carnegie Mellon University, 2005, CMU-LTI-05-192
150. X. Zhu, Semi-supervised learning literature survey (2006), Available online at http://pages.cs.wisc.edu/~jerryzhu
151. X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002
152. J. Zhu, S. Rosset, T. Hastie, R. Tibshirani, 1-norm support vector machines, in *Proceedings of Advances in Neural Information Processing Systems*, Vancouver, 2003
153. X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in *Proceedings of 21st International Conference on Machine Learning (ICML)*, Washington, DC, USA, vol. 20, 2003, p. 912
154. H. Zou, M. Yuan, The $f_\infty$-norm support vector machine. Stat. Sin. **18**, 379–398 (2008)