EasyChair Preprint

№ 1277

Support Vector Machine with Graphical Network Structures in Features

Wenqing He, Grace Yi and Li-Pang Chen

July 10, 2019

Support Vector Machine with Graphical Network Structures in Features

Wenqing He[1], Grace Y. Yi[2], and Li-Pang Chen[3]

[1] University of Western Ontario, London, Ontario, Canada
whe@stats.uwo.ca
[2] University of Waterloo, Waterloo, Ontario, Canada
yyi@uwaterloo.ca
[3] University of Waterloo, Waterloo, Ontario, Canada
l358chen@uwaterloo.ca

Abstract. Machine learning techniques, regardless of being *supervised* or *unsupervised*, have attracted extensive research attention in handling data classification. Typically, among supervised machine learning algorithms, Support Vector Machine (SVM) and its extensions have been widely used in various areas due to their great prediction capability. These learning algorithms basically treat features of the instances independently when using them to do classification. However, in applications, features are commonly correlated with complex network structures. Ignoring such a characteristic and naively implementing the SVM algorithm may yield erroneous classification results. To address the limitation of the SVM algorithm, we propose new learning algorithms which accommodate network structures in the features of the instances. Our algorithms capitalize on graphical model theory and make use of the available R software package for SVM. The implementation of the proposed learning algorithms is computationally straightforward. We apply the new algorithms to analyze the data arising from a gene expression study.

Keywords: Classification · Graphical model · Network structure · Support vector machine.

1 Introduction

In supervised learning, the support vector machine (SVM) classifier has been popularly used. Initially proposed for binary classification, the SVM has a number of merits including accuracy and robustness for prediction (e.g., [2]), [3], [23]. The SVM has been successfully applied to different areas, including genomic studies in medical science [8], text classification [11], and image retrieval [22].

Various extensions of the SVM have been available. For instance, the least squares support vector machine (LS-SVM) and its variant, the weighted LS-SVM, were developed. These methods work with (weighted) least squared cost functions and impose equality constraints instead of inequality constraints required by the standard SVM method ([19], [20]). Other useful variants of the SVM include the bagging and boosting methods which ensemble individual classifiers to further enhance the accuracy of the classification procedures (e.g., [12], [16]).

Although different extensions of the SVM have been proposed, research gaps still remain. Notably, almost all available methods treat the features of the instances independently and ignore possible dependence structures among them. In applications, however, data are often correlated in various forms. For example, in genomic studies some genes may be associated since they come from the same pathway, and they may function together in the development of diseases. Ignoring the complex association structures of the features in instances and blindly using the SVM or its variants for classification can yield misleading or erroneous results. Driven by this, in this paper we develop new SVM-based classification methods with association network structures among the features incorporated. Our methods utilize the graphical model theory and allow the features to follow an exponential family distribution. Our algorithms facilitate informative features with complex network structures into classification learning procedures, and they are computationally easy to implement.

Our research is partially motivated by tackling the challenges arising from a gene expression study [7]. The study targets to identify gene signature for the distinction between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). The primary objectives are to identify possible pathways of genes that are expressed together, and to classify future patients into the AML or ALL group for the effective treatment by using the identified gene signature. To this end, it is important to understand the association structures among the genes to effectively classify the classes of AML and ALL for prediction.

The remainder of this article is organized as follows. In Section 2, we introduce the framework and review the SVM method. Discussion on identifying the network structures of the features is given in Section 3. In Section 4, we propose new learning algorithms which generalize the scope of the support vector machine classifiers. In Section 5, we apply the proposed algorithms to analyze the gene expression dataset from the motivating microarray study [7]. A general discussion is presented in the last section.

2 Background and Review of Support Vector Machine

Consider a binary classification problem with a training sample $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, where $\mathbf{x}_i$ represents a p-dimensional column vector of the features for instance i, and y_i is the corresponding class label for instance i, taking a value from {-1, 1} which stands for two different classes. Given the training data, the standard SVM finds a hyperplane

$$\{\mathbf{x} \in \mathbb{R}^p : \beta'\mathbf{x} + b_0 = 0\} \tag{1}$$

that separates the two classes with a maximized margin defined by $\frac{y(\beta'\mathbf{x}+b_0)}{\|\beta\|_2}$, where β is a p-dimensional column parameter vector determining the direction of the separating hyperplane, b_0 is a scalar which measures the shift of the plane from the origin, $(\mathbf{x}, y)$ is an observation with instance features $\mathbf{x}$ and label y, and $\|\cdot\|_2$ represents the L_2 norm.

Equivalently, using the training sample $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ to find (1) with the maximal margin can be carried out by minimizing $\|\beta\|_2$ under certain constraints [9],

i.e.,

$$\min_{\beta, b_0} \quad \left\{ \frac{1}{2}\, \beta'\beta \right\} \tag{2}$$

subject to

$$y_i\{\beta'\mathbf{x}_i + b_0\} \geq 1 \qquad \text{for } i = 1, \ldots, n.$$

The solution of the optimization (2) gives a band of $2 \times \frac{1}{\|\beta\|_2}$ unit between the two classes, called the *margin area*, with the hyperplane $\beta'\mathbf{x} + b_0 = 0$ being in the middle of the band. Such a determined hyperplane separates the two classes perfectly with a minimal distance of $2 \times \frac{1}{\|\beta\|_2}$ between the two classes, and does not allow any instance to fall into the interior of the margin area. In applications, however, such a hyperplane may not exist to perfectly separate the instances into each side according to their classes.

One remedy is to relax the requirement of perfectly separating the instances in the two classes and allow some instances to fall into the margin area associated with a hyperplane or even across to the wrong side of the hyperplane. To the end, we introduce the so-called *slack variable*, say ξ_i, for $i = 1, \ldots, n$, to indicate the margin error for each instance. Then the optimization problem (2) is relaxed as

$$\min_{\beta, b_0, \boldsymbol{\xi}} \quad \left\{ \frac{1}{2}\, \beta'\beta \; + \; B \sum_{i=1}^{n} \xi_i \right\} \tag{3}$$

subject to

$$y_i\{\beta'\mathbf{x}_i + b_0\} \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \qquad \text{for } i = 1, \ldots, n,$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)'$, B is a cost parameter controlling the balance between maximizing the margin and minimizing the training error caused by allowing some instances to fall into the margin or cross the separating boundary. This procedure is termed the *soft margin* SVM.

Using the Lagrangian multiplier method, the minimization problem (3) can be rewritten as

$$\min_{\beta, b_0, \boldsymbol{\xi}} \quad \left(\frac{1}{2}\, \beta'\beta \; + \; B \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \Big[y_i\{\beta'\mathbf{x}_i + b_0\} - 1 + \xi_i \Big] - \sum_{i=1}^{n} \mu_i \xi_i \right), \tag{4}$$

where for $i = 1, \ldots, n$, α_i and μ_i, satisfying $0 \leq \alpha_i \leq B$ and $\mu_i \geq 0$, are the Lagrangian multipliers corresponding to the inequality constraints in (3), and $\xi_i \geq 0$.

The procedure (4) is a standard convex optimization problem which can be equivalently implemented using its dual formulation:

$$\max_{\boldsymbol{\alpha}} \quad \left\{ \sum_{i=1}^{n} \alpha_i \; - \; \frac{1}{2} \sum_{i,\, j} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j > \right\} \tag{5}$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq B \qquad \text{for } i = 1, \ldots, n,$$

where $< \cdot, \cdot >$ represents the inner product of two vectors, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)'$ is the vector of constants (e.g., [1], Section 13.2).

Due to the linearity of the function $\beta'\mathbf{x} + \mathbf{b_0}$ in (1), the learning algorithm (4) is easy to implement. But there is an important drawback. In many situations, even with some instances allowed to be misclassified, one cannot obtain a hyperplane of the form (1) to reasonably separate the classes. That is, a *linear* hyperplane is not adequate to separate the majority of data. To handle this problem, we enlarge the framework of classification by embedding the data with p-dimensional features into a new space (denoted $\mathscr{F}$) with a higher dimension q, say, so that a hyperplane can be constructed in the new space to separate the classes [3].

To be precise, let $\Phi(\mathbf{x})$ denote a projection function which maps the original feature $\mathbf{x}$ (in the p-dimensional feature space, say $\mathscr{X}$) into the q-dimensional projected space $\mathscr{F}$. Then we repeat the preceding SVM procedures to find a hyperplane in the projected space $\mathscr{F}$:

$$\mathbf{w}'\Phi(\mathbf{x}) + b = 0$$

which separates the two classes with a maximum margin in the projected space $\mathscr{F}$, where $\mathbf{w}$ is a q-dimensional column vector determining the direction of the separating hyperplane in the projected space $\mathscr{F}$, and b is a scalar which measures the shift of the plane from the origin. The corresponding Lagrangian formulation can be obtained in the same manner as (4) with $\mathbf{x}_i$ replaced by $\Phi(\mathbf{x}_i)$.

Such a procedure requires an explicit form of the projection function $\Phi(\cdot)$ which is basically unavailable. An improper specification of $\Phi(\cdot)$ may seriously deteriorate the performance of the support vector machine [9]. Moreover, the projection function $\Phi(\cdot)$ is often nonlinear, and the number of its evaluation for the observations increases with the size n of the training data.

To overcome these issues, instead of finding a suitable function $\Phi(\cdot)$, a viable strategy is to introduce the *kernel function* and then reformulate (5) as

$$\max_{\boldsymbol{\alpha}} \quad \left\{ \sum_{i=1}^{n} \alpha_i \; - \; \frac{1}{2} \sum_{i,\, j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right\} \tag{6}$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \le \alpha_i \le B \quad \text{for } i = 1, \ldots, n,$$

where $K(\cdot, \cdot)$ is a kernel function whose value depends only on the inner product of the transformations of $\mathbf{x}_i$ and $\mathbf{x}_j$ in the projected space $\mathscr{F}$ [13], B is the cost parameter, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)'$ is the vector of constants (e.g., [1], Section 13.2).

The solution of the quadratic optimization problem (6) is optimal if and only if the kernel function $K(\cdot, \cdot)$ satisfies the Mercer's condition and the Karush-Kuhn-Tucker conditions [9]. Basically, only those instances $\mathbf{x}_i$ with non-zero α_i are useful in determining the hyperplane, and they are called the *support vectors* [4]. Geometrically, the support vectors correspond to the instances on the edges, those entering into the margin area, and those crossing the wrong side of the separating hyperplane in the projected space $\mathscr{F}$.

Common choices of kernel functions include the polynomial kernel, $K(\mathbf{x}_i,\mathbf{x}_j) = (1+\mathbf{x}_i'\mathbf{x}_j)^d$; the Gaussian kernel,

$$K(\mathbf{x}_i,\mathbf{x}_j) = \exp\left\{-\frac{(\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{x}_i-\mathbf{x}_j)}{\sigma}\right\};$$

and the sigmoidal kernel, $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(2\mathbf{x}_i'\mathbf{x}_j+1)$; where d is a positive integer and σ is a positive scalar parameter (e.g., [1], Section 13.6).

The SVM is conceptually easy to understand and computationally manageable. Its solution enjoys the sparsity in the sense that the number of support vectors is considerably smaller than the size of the training data. The flexibility of the inclusion of kernel functions allows us to use the SVM to classify nonlinearly separable data. Due to the uniqueness of the solution, there is no need to invoke any iterative optimization procedure as commonly needed for other classification methods such as neural networks. The SVM method is considered to be the best learning algorithm and is popularly used in many areas (e.g., [1]).

3 Feature Network Structure

Although the SVM algorithm has been widely used, it has limitations. A notable weakness of the SVM method is the ignorance of the structures of features. While using different kernel functions gives the SVM classification flexibility in separating data, this learning algorithm is restrictive in that the features of the instances are all equally treated and there is no distinction between important and unimportant features; the relationship of the features is essentially ignored. When the features of the instances exhibit complex network structures, the usual SVM algorithm with a direct use of kernel functions may fail for proper classification.

To handle data with structures existing in the features, in this paper we develop new classification learning algorithms which accommodate the usual SVM method as a special case. Our learning algorithms consist of two steps where in the first step, we explore the network structure of the features using the graphical theory, as described in this section.

3.1 Basics of Graphical Models

We use an undirected *graph*, denoted as $G_i = (V_i, E_i)$, to describe the relationship among the features of the instance $\mathbf{x}_i = (x_{i1},\cdots,x_{ip})'$, where we let V_i denote the set of all the indexes of the features in $\mathbf{x}_i$ and let E_i denote the subset of $V_i \times V_i$ which contains the index pairs for those features in X_i that are related, in the sense to be elaborated in the sequel. We consider the case where the sets V_i and E_i are common for $i = 1,\cdots,n$, so we let V and E denote them, respectively.

To characterize the distribution of the instance $\mathbf{x}_i$, we consider the exponential family distribution

$$f(\mathbf{x}_i;\beta,\Theta) = \exp\left\{\sum_{r\in V}\beta_r B(x_{ir}) + \sum_{(s,t)\in E}\theta_{st}B(x_{is})B(x_{it}) + \sum_{r\in V}C(x_{ir}) - A(\beta,\Theta)\right\}, \quad (7)$$

where $\beta = (\beta_1, \cdots \beta_p)'$ is the p-dimensional vector of parameters, $\Theta = [\theta_{st}]$ is a $p \times p$ symmetric matrix with diagonal elements zeros, $B(\cdot)$ and $C(\cdot)$ are given functions, and $A(\beta, \Theta)$ is the normalizing constant which makes (7) integrated as 1.

Formulation (7) gives a broad class of models which covers many useful distributions. For example, if $B(u) = \frac{u}{\sigma}$ and $C(u) = -\frac{u^2}{2\sigma^2}$ for a positive parameter σ, then (7) yields the well-known *Gaussian graphical model* (e.g, [6], [10], [14]). If setting $\beta = 0$, $B(u) = u$, and $C(u) = 0$ with $u \in \{0, 1\}$, then (7) reduces to

$$\exp\left\{\sum_{(s,t)\in E} \theta_{st} x_{is} x_{it} - A(\Theta)\right\}, \tag{8}$$

which is the *Ising model* without the singletons [18].

To focus on featuring the pairwise association among the features of x_i, similar to the structure of (8), we consider the distribution

$$f(x_i; \Theta) = \exp\left\{\sum_{(s,t)\in E} \theta_{st} x_{is} x_{it} + \sum_{r\in V} C(x_{ir}) - A(\Theta)\right\}, \tag{9}$$

where the function $A(\Theta)$ is the normalizing constant, and the θ_{st} and $C(\cdot)$ are defined as for (7). Model (9) is a special case of (7) which constraints the main effects parameters β_r in (7) to be zero; a nonzero value of θ_{st} implies that the features X_{is} and X_{it} are conditionally dependent given other features.

3.2 Learning Model Parameters

To learn the value of Θ, one may apply the likelihood method using the distribution (9) directly. Alternatively, a simpler learning method can be carried out based on a conditional distribution derived from (9) ([10], [17]). For every $s \in V$, let $x_{i,V\setminus\{s\}}$ denote the $(p-1)$-dimensional subvector of x_i with its sth component deleted, i.e., $x_{i,V\setminus\{s\}} = (x_{i1}, \cdots, x_{i,s-1}, x_{i,s+1}, \cdots, x_{ip})'$. By some algebra, we have

$$f\left(x_{is} | x_{i,V\setminus\{s\}}; \theta_s\right) = \exp\left\{x_{is}\left(\sum_{t\in V\setminus\{s\}} \theta_{st} x_{it}\right) + C\left(x_{is}\right) - D\left(\sum_{t\in V\setminus\{s\}} \theta_{st} x_{it}\right)\right\}, \tag{10}$$

where $D(\cdot)$ is the normalizing function ensuring the integration of (10) equal one, and $\theta_s = (\theta_{s1}, \cdots, \theta_{s,s-1}, \theta_{s,s+1}, \cdots, \theta_{sp})'$ is a $(p-1)$-dimensional vector of parameters indicating the relationship of X_{is} with all other features X_{ir} for $r \in \{1, \cdots, p\} \setminus \{s\}$.

Let $\ell(\theta_s)$ be the log likelihood for θ_s obtained from using (10) with the multiplication factor $-\frac{1}{n}$ included and the parameter-free terms omitted:

$$\ell(\theta_s) = \; = \frac{1}{n}\sum_{i=1}^{n}\left\{-x_{is}\left(\sum_{t\in V\setminus\{s\}} \theta_{st} x_{it}\right) + D\left(\sum_{t\in V\setminus\{s\}} \theta_{st} x_{it}\right)\right\}. \tag{11}$$

Then learn the value of θ_s by penalizing the log likelihood (11), given by

$$\widehat{\theta}_s = \underset{\theta_s}{\operatorname{argmin}} \left\{\ell(\theta_s) + \lambda \left\|\theta_s\right\|_1\right\},$$

where λ is a tuning parameter, and $\|\cdot\|_1$ is the L_1-norm, or the well-known LASSO penalty [21]. The LASSO penalty is frequently considered when dealing with graphical models; it has been implemented in R with the available packages "huge" and "XMRF".

While the choice of the tuning parameter λ is not unique, we employ the BIC approach here to select the tuning parameters λ, as suggested by [24]. BIC tends to outperform other criteria in many situations, especially in the setting with a penalized likelihood function.

The preceding procedure is repeated for all $s \in V$ and yields the learned value $\widehat{\theta}_s$ for all $s \in V$. For $(s,t) \in E$, the learned $\widehat{\theta}_{st}$ and $\widehat{\theta}_{ts}$ are not necessarily identical although θ_{st} and θ_{ts} are constrained to be equal. To overcome this problem, we apply the AND rule ([10], [17]), which takes the final learned values of $\widehat{\theta}_{st}$ and $\widehat{\theta}_{ts}$ to be their maximum if both $\widehat{\theta}_{st}$ and $\widehat{\theta}_{ts}$ are not equal to zero; $\widehat{\theta}_{st}$ and $\widehat{\theta}_{ts}$ are both set to be zero if one of them is zero.

To determine a learned set of edges, we define

$$\widehat{\mathscr{N}}(s) = \left\{ t \in V : \widehat{\theta}_{st} \neq 0 \right\}$$

for $s \in V$. Then

$$\widehat{E} = \left\{ (s,t) : s \in \widehat{\mathscr{N}}(t) \text{ and } t \in \widehat{\mathscr{N}}(s) \right\}$$

is taken as the set of the edges that are learned to exist. The R package "huge" can be implemented to display the graphic results.

Under regularity conditions, the learned set of edges $\widehat{E}$ approximates the true network structure E with a high probability. [17] showed that $P\left(\widehat{E} = E\right) \to 1$ as $n \to \infty$.

4 Support Vector Machine with Network Structured Features

In this section, we propose two new learning algorithms which generalize the support vector machine classifier to incorporate network structured features. These algorithms differ in the way of learning the network structures of the features in the space $\mathscr{X}$.

4.1 SVM with Homogenous Network Structured Features

The first learning algorithm applies to the case where the feature space $\mathscr{X}$ contains homogeneous structures for all the classes. In this case, we ignore the class label and pool the instances when using the training data to learn the feature network.

To be precise, in the training data $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, we use the measurements $\{\mathbf{x}_i : i = 1, \ldots, n\}$ for the p-dimensional features $\mathbf{x} = (x_1, \ldots, x_p)'$ to identify the network structure among the features $x_k (k = 1, \ldots, p)$. Using the learning algorithm described in Section 3, we learn a network structure for the p features:

$$\widehat{E}_{\text{P}} = \left\{ (s,t) : s \in \widehat{\mathscr{N}_{\text{P}}}(t) \text{ and } t \in \widehat{\mathscr{N}_{\text{P}}}(s) \right\}, \tag{12}$$

where

$$\widehat{\mathscr{N}_{\text{P}}}(s) = \left\{ t \in V : \widehat{\theta}_{st} \neq 0 \right\} \tag{13}$$

for $s \in V$. We call the identified feature structure $\widehat{E}_{\text{P}}$ the *homogenous network structure*.

To reflect different association structures among the features x_k $(k = 1, \ldots, p)$, we divide the estimated graph $\widehat{G}_{\text{P}} = (V, \widehat{E}_{\text{P}})$ as a sequence of non-overlapped and interconnected subgraphs $\{\widehat{G}^s_{\text{P}} : s = 1, ..., K\}$, where K is a positive integer. That is, for $s = 1, \ldots, K$, $\widehat{G}^s_{\text{P}} = (\widehat{V}^s, \widehat{E}^s)$ is a subgraph with the vertex subset $\widehat{V}^s$ and the edge subset $\widehat{E}^s$; the vertex sets $\widehat{V}^s$ and $\widehat{V}^t$ are disjoint if $s \neq t$; $\widehat{\theta}_{st}$ is not zero only when s and t fall in the same vertex subset, and for s and t which fall in different vertex subsets, $\widehat{\theta}_{st}$ must be zero; when the edge subset $\widehat{E}^s$ is empty, the corresponding vertex subset $\widehat{V}^s$ contains a single element. In principle, the integer K can be as small as 1 and as large as p, but in many practical problems, we often have that $1 < K < p$ and that K can be considerably much smaller than p.

The subgraph sequence $\{\widehat{G}^s_{\text{P}} : s = 1, ..., K\}$ can not only display the network structures of the features x_k $(k = 1, \ldots, p)$ but also offers us an effective way to summarize the associated feature information, thus, achieving better classification learning outcomes. Rather than using the original measurements of the training data $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ to classify the class labels, we make use of the network structures to form a *surrogate* training sample $\{(\mathbf{x}^*_i, y_i) : i = 1, \ldots, n\}$ where $\mathbf{x}^*_i = (x^{*'}_{i1}, \ldots, x^{*'}_{iK})'$, called the *surrogate features*, is a K-dimension vector, and x^*_{is} is a summary subvector of the features determined by the subgraph $\widehat{G}^s_{\text{P}}$ for $s = 1, \ldots, K$.

While multiple ways are possible to define the *surrogate features* $\mathbf{x}^*_i$ for each instance i, here we consider three useful formulations. The first formulation summarizes the feature measurements using the vertex information in the subgraphs and defines $\mathbf{x}^*_i$ to be $\mathbf{x}^{\text{V}}_i = (x^{\text{V}}_{i1}, \ldots, x^{\text{V}}_{iK})'$ with

$$x^{\text{V}}_{is} = \frac{1}{|\widehat{V}^s|} \sum_{j \in \widehat{V}^s} x_{ij}$$

for $s = 1, \ldots, K$, where $|\widehat{V}^s|$ is the cardinality of the vertex subset $\widehat{V}^s$.

The second formulation uses the edge information in the subgraphs and defines $\mathbf{x}^*_i$ to be $\mathbf{x}^{\text{E}}_i = (x^{\text{E}}_{i1}, \ldots, x^{\text{E}}_{iK})'$ with

$$x^{\text{E}}_{is} = \frac{1}{|\widehat{E}^s|} \sum_{(j,k) \in \widehat{E}^s} x_{ij} x_{ik}$$

for $s = 1, \ldots, K$, where $|\widehat{E}^s|$ is the cardinality of the edge subset $\widehat{E}^s$; when $\widehat{E}^s$ is empty, we define x^{E}_{is} to be the feature x_{is} whose index falls in the corresponding vertex subset $\widehat{V}^s$.

There is certain similarity of the formulation of $\mathbf{x}^{\text{V}}_i$ to the k-nearest neighbors (k-NN) algorithm, but the "neighborhood" in defining $\mathbf{x}^{\text{V}}_i$ is not determined by a distance of instances but determined by the association strength of the features. The surrogate features $\mathbf{x}^{\text{V}}_i$ basically are the average of original measurements of the features that are determined by the vertex subsets of the network structures $\widehat{G}$, and each component in the vertex subsets is treated equally. In contrast, the surrogate features $\mathbf{x}^{\text{E}}_i$ accommodate pairwise association among the features, and a feature which is more connected with other features contributes differently from a feature less connected with others.

Extensions of the formulation of $\mathbf{x}_i^{\text{V}}$ or $\mathbf{x}_i^{\text{E}}$ can also be employed for individual applications. For example, one may introduce weights in the formulation to reflect different contributions of different features in conjunction of the feature network structure. In situations with outliers, rather than taking the average operation, one may use the median of $\{x_{ij} : j \in \widehat{V}^s\}$ or $\{x_{ij}x_{ik} : (j,k) \in \widehat{E}^s\}$ to define x_{is}^{V} or x_{is}^{E}.

A third method of constructing the surrogate feature $\mathbf{x}_i^* = (x_{i1}^*, ..., x_{ik}^*)'$ is to let each subvector x_{is}^* be the vector $(x_{ij}x_{ik} : (j,k) \in \widehat{E}_s)'$, which corresponds to the pairwise product for those connected features in subgraph $\widehat{G}_s$, where $s = 1, ..., K$. we let $\mathbf{x}_i^{\text{PW}}$ denote such a surrogate feature vector.

Once a surrogate training sample $\{(\mathbf{x}_i^*, y_i) : i = 1, \ldots, n\}$ is formed, in the next step we apply the SVM algorithm (6) with modifications to perform classification. Namely, for a chosen kernel function $K^*(\cdot,\cdot)$, solve the constrained optimization

$$\max_{\boldsymbol{\alpha}^*} \quad \left\{ \sum_{i=1}^{n} \alpha_i^* \; - \; \frac{1}{2} \sum_{i,\; j} \alpha_i^* \alpha_j^* y_i y_j K^*(\mathbf{x}_i^*, \mathbf{x}_j^*) \right\} \tag{14}$$

subject to

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0 \quad \text{and} \quad 0 \le \alpha_i^* \le B^* \quad \text{for } i = 1, \ldots, n,$$

where $\boldsymbol{\alpha}^* = (\alpha_1^*, \ldots, \alpha_n^*)'$ is the vector of constants, and B^* is a cost parameter.

4.2 SVM with Heterogeneous Network Structured Features

The second learning algorithm complements the algorithm described in Section 4.1. This algorithm applies when the feature space $\mathscr{X}$ includes nonhomogeneous structures for different classes. In such a circumstance, it is imperative to incorporate the class labels into the process of learning the feature network structures.

Let $N_{+1} = \{i \in \{1, ..., n\} : y_i = 1\}$ and $N_{-1} = \{i \in \{1, ..., n\} : y_i = -1\}$ denote the index sets for the instances in the two classes. The algorithm consists three steps. First, we examine the network structures for the p-dimensional features $\mathbf{x} = (x_1, \ldots, x_p)'$ separately according to the class label. Secondly, we use the two sets of surrogate samples obtained from the first step to construct two hyperplanes separately, and finally, we carry out prediction for future instances based on these two hyperplanes.

To be precise, we proceed with the following three steps:

Step 1: We identify network structures for instances.

For the training data subsample with instances $\{\mathbf{x}_i : i \in N_{+1}\}$, we employ the same learning algorithm described in Section 3 except for replacing $i = 1, \ldots, n$ with $i \in N_{+1}$, and then we obtain a learned network structure for the p features that are associated with the class label $+1$:

$$\widehat{E}_{+1} = \left\{ (s,t) : s \in \widehat{\mathscr{N}}_{+1}(t) \text{ and } t \in \widehat{\mathscr{N}}_{+1}(s) \right\},$$

where

$$\widehat{\mathscr{N}}_{+1}(s) = \left\{ t \in V : \widehat{\theta}_{st}^{+1} \neq 0 \right\}$$

for $s \in V$, and $\widehat{\theta}_{st}^{+1}$ is the learned value of θ_{st} using instances $\{\mathbf{x}_i : i \in N_{+1}\}$.

Similarly, repeating the algorithm for the training data subsample with instances $\{\mathbf{x}_i : i \in N_{-1}\}$, we learn a network structure for the features associated with the class label -1:

$$\widehat{E}_{-1} = \left\{(s,t) : s \in \widehat{\mathscr{N}_{-1}}(t) \text{ and } t \in \widehat{\mathscr{N}_{-1}}(s)\right\},$$

where

$$\widehat{\mathscr{N}_{-1}}(s) = \left\{t \in V_{-1} : \widehat{\theta}_{st}^{-1} \neq 0\right\}$$

for $s \in V$, and $\widehat{\theta}_{st}^{-1}$ is the learned value of θ_{st} based on instances $\{\mathbf{x}_i : i \in N_{-1}\}$. We call the identified feature structures $\widehat{E}_{+1}$ and $\widehat{E}_{-1}$ the *label heterogeneous network structures*.

Step 2: We determine two hyperplanes using the label heterogenous network structures $\widehat{E}_{+1}$ and $\widehat{E}_{-1}$.

For the learned graph $\widehat{G}_{+1} = (V, \widehat{E}_{+1})$ corresponding to the class with label $+1$, we divide it as a sequence of non-overlapped and inter-connected subgraphs $\{\widehat{G}_{+1}^s : s = 1, ..., K_{+1}\}$ following the discussion in Section 4.1. Based on them, we form a set of surrogate features for all the instances, and let $\{(\mathbf{x}_{i,+1}^*, y_i) : i = 1, ..., n\}$ denote the resulting surrogate sample, where $\mathbf{x}_{i,+1}^*$ can be determined similarly to $\mathbf{x}_i^{\text{V}}$, $\mathbf{x}_i^{\text{E}}$ or $\mathbf{x}_i^{\text{PW}}$ as discussed in Section 4.1.

Then we apply the optimization (14) to the surrogate training data $\{(\mathbf{x}_{i,+1}^*, y_i) : i = 1, ..., n\}$, and obtain the separating hyperplane

$$f_{+1}(\mathbf{x}_{+1}^*) = 0$$

where

$$f_{+1}(\mathbf{x}_{+1}^*) = \sum_{i=1}^{n} \widehat{\alpha}_i^+ y_i K^+(\mathbf{x}_{+1}^*, \mathbf{x}_{i,+1}^*) + b_{+1}, \tag{15}$$

$\mathbf{x}_{+1}^*$ is obtained from applying the subgraph structure $\hat{G}_{+1}$ to a given feature vector $\mathbf{x}$, $K^+(\cdot,\cdot)$ is a kernel function, and $\widehat{\alpha}_i^+$ and b_+ are, respectively, the learned values of α_i and b using the surrogate training sample $\{(\mathbf{x}_{i,+1}^*, y_i) : i = 1, ..., n\}$.

In an analogous way, for the estimated graph $\widehat{G}_{-1} = (V, \widehat{E}_{-1})$ corresponding to the class with label -1, we divide it as a sequence of non-overlapped and inter-connected subgraphs $\{\widehat{G}_{-1}^s : s = 1, ..., K_{-1}\}$, and form a surrogate sample $\{(\mathbf{x}_{i,-1}^*, y_i) : i = 1, ..., n\}$, where $\mathbf{x}_{i,-1}^*$ is obtained similarly to $\mathbf{x}_{i,+1}^*$.

Repeating the procedure for obtaining (15) to the surrogate training data $\{(\mathbf{x}_{i,-1}^*, y_i) : i = 1, ..., n\}$, we obtain the separating hyperplane

$$f_{-1}(\mathbf{x}_{-1}^*) = 0$$

where

$$f_{-1}(\mathbf{x}_{-1}^*) = \sum_{i=1}^{n} \widehat{\alpha}_i^- y_i K^-(\mathbf{x}_{-1}^*, \mathbf{x}_i^*) + b_{-1}, \tag{16}$$

$\mathbf{x}^*_{-1}$ is obtained from applying the subgraph structure $\hat{G}_{-1}$ to the feature vector $\mathbf{x}$, $K^-(\cdot,\cdot)$ is a kernel function, and $\widehat{\alpha}_i^{-1}$ and b_- are, respectively, the learned values of α_i and b using the surrogate training sample $\{(\mathbf{x}^*_{i,-1}, y_i) : i = 1, ..., n\}$.

Step 3: We perform prediction.

To predict the class label for a new instance $\mathbf{x}$, we first construct the surrogate feature vectors $\mathbf{x}^*_{+1}$ and $\mathbf{x}^*_{-1}$, respectively, based on $\hat{G}_{+1}$ and $\hat{G}_{-1}$, and then calculate $f_{+1}(\mathbf{x}^*_{+1})$ and $f_{-1}(\mathbf{x}^*_{-1})$ using (15) and (16), respectively. Finally, the class label for the instance $\mathbf{x}$ is assigned as $+1$ if

$$|f_{+1}(\mathbf{x}^*_{+1})| > |f_{-1}(\mathbf{x}^*_{-1})| \tag{17}$$

and -1 otherwise.

5 Real Data Application

Golub et al. [7] reported a gene expression microarray analysis expecting to identify gene signature for the distinction between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL), where gene expression levels were measured using Affymetrix oligonucleotide arrays. The data contain 7128 genes and 72 specimens coming from the two classes, with 47 specimens in class ALL and 25 specimens in class AML. In particular, according to the study design, those 72 samples is composed of the training data of 38 specimens (27 in class ALL and 11 in class AML) and the test data of 34 specimens (20 in class ALL and 14 in class AML). The primary objectives are (1) to identify the genes that are expressed differentially between AML and ALL, (2) to find possible pathways of genes that are expressed together, and (3) to classify the classes of AML and ALL using the gene expressions. We apply the proposed SVM learning algorithms to incorporate the network structures of the genes (as possible pathway effects) to do prediction of the leukemia class.

Since the number of genes is large relative to the sample size, it is necessary to do feature screening to remove non-informative features before applying a learning algorithm. While various feature screening methods available in the literature (e.g., [5], [10], [15],), we choose the feature screening method in [15] because it is free of model specification and is available in the R package `energy`. It turns out that 14 genes are selected from the 38 training sample data, which are strongly correlated to the response of different leukemia types, and we labeled them by $1, 2, \cdots, 14$.

We now apply the proposed learning algorithms to these 14 selected genes. In the first step, we use the package "`XMRF`" to determine the network structures of the 14 genes separately for class ALL and class AML. The learned network structures are displayed in Figure 1. In contrast, we also learn network structure by pooling the samples for the two class, and the homogeneous network structure is displayed in Figure 2. We apply the three methods described in Section 4 to construct surrogate features. Gaussian radial kernel is used in the SVM classifiers. For comparison purposes, the conventional SVM model is implemented using the R package "`e1071`".

We then apply all the classifiers learned from the training sample to the 34 test specimens to examine their performance. The prediction results are shown in Table 1. All the classifiers yield comparable results, and the proposed algorithms perform slightly better than the conventional SVM algorithm. Furthermore, we evaluate the performance of the classifiers using three widely used measures, *precision, recall*, and *F-measure*, and report on the results in Table 2. It is evident that the proposed classifiers all outperform the standard SVM algorithm, regardless how we learn the network structures for the features. Applying the new learning algorithms enable us to offer a better prediction of the leukemia classes than the traditional SVM does. In addition, it gives us descriptions of the association structures of the features, which is important to be incorporated in the classification process.

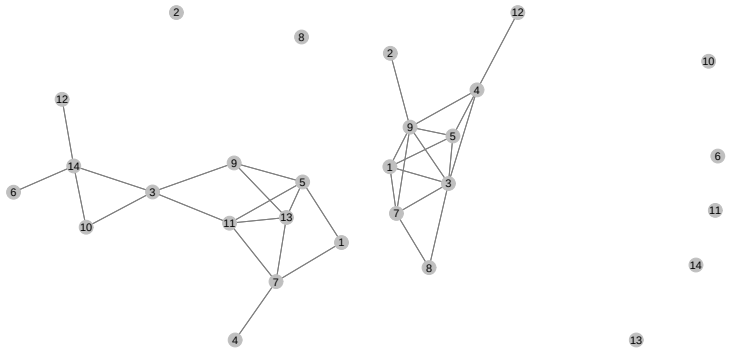

Fig. 1. Network structures of the features within each class.

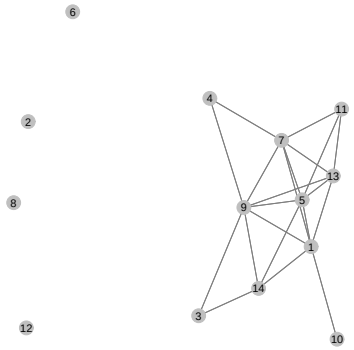

Fig. 2. Network structures of the features for pooled classes.

Table 1. 2×2 contingency table for prediction

	Heterogeneous-Network						Homogeneous-Network						SVM	
	vertex		edge		pair		vertex		edge		pair			
	ALL	ALM	ALL	ALM	ALL	ALM	ALL	ALM	ALL	ALM	ALL	ALM	ALL	ALM
ALL	14	1	13	0	13	0	14	1	13	0	13	0	13	1
ALM	0	19	1	20	1	20	0	19	1	20	1	20	1	19

Table 2. Performance of the new learning algorithms in comparison to SVM

	Heterogeneous-Network			Homogeneous-Network			SVM
	vertex	edge	pair	vertex	edge	pair	
Precision	0.933	1.000	1.000	0.993	1.000	1.000	0.928
Recall	1.000	0.928	0.928	1.000	0.928	0.928	0.928
F-measure	0.965	0.963	0.965	0.954	0.965	0.965	0.928

6 Discussion

Classification is of broad interest in many fields, and features used for prediction are often correlated with network structures. Ignoring association structures of features may produce unsatisfactory prediction outcomes. In this paper, we employ the graphical model theory to identify network structures for the features, and then incorporate such a characteristic into the classification learning process. The application of our proposed algorithms demonstrates their better performance than the usual SVM algorithm.

As described in Section 4, construction of surrogate features is not unique. In our application, we explore three ways of summarizing the information carried by the feature networks. Other methods such as weight mean statistics or weighted robust statistics may also be considered, and it is worthwhile to examine how they affect learning outcomes.

Acknowledgements

The authors thank the reviewers for their comments on the initial version. The research is supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and partially supported by the Canadian Statistical Sciences Institute (CANSSI).

References

1. Alpaydyn, E.: Introduction to MachineLearning. The MIT Press, Cambridge, 2nd edn. (2010)

2. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning **46**, 131–159 (2002)
3. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**, 273–297 (1995)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, New York (2000)
5. Fan, J., Li, R.: Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the Americal Statistical Association **96**, 1348–1360 (2001)
6. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**, 432–441 (2008)
7. Golub, L., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., Lander, E.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science **286**, 531–537 (1999)
8. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46**, 389–422 (2002)
9. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Prediction, Inference, and Data Mining. Springer, Berlin, 2nd edn. (2009)
10. Hastie, T., Tibshirani, R., Wainwright, M.: Statistical Learning with Sparsity: The Lasso and Generalizations. CRC press, New York (2015)
11. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: the Tenth European Conference on Machine Learning. pp. 137–142 (1998)
12. Kim, H., Pang, S., Je, H., Kim, D., Bang, S.: Support vector machine ensemble with bagging. Pattern Recognition with Support Vector Machines pp. 397–408 (2002)
13. Kumar, S., Hebert, M.: Discriminative random fields. International Journal of Computer Version **68**, 179–201 (2006)
14. Lee, J.D., Hastie, T.J.: Learning the structure of mixed graphical models. J Comput Graph Stat. **24**, 230–253 (2015)
15. Li, R., Zhong, W., Zhu, L.: Feature screening via distance correlation learning. Journal of the American Statistcial Association **107**, 1129–1139 (2012)
16. Liu, Y., An, A., Huang, X.: Boosting prediction accuracy on imbalanced datasets with svm ensembles. Pacific-Asia Conference on Knowledge Discovery and Data Mining: Advances in Knowledge Discovery and Data Mining pp. 107–118 (2006)
17. Meinshausen, N., Buhlmann, P.: High-dimensional graphs and variable selection with the lasso. The Annals of Statistics **34**, 1436–1462 (2006)
18. Ravikumar, P., Wainwright, M.J., Lafferty, J.: High-dimensional ising model selection using l1-regularized logistic regression. The Annals of Statistics **38**, 1287–1319 (2010)
19. Suykens, J., De Brabanter, J., Lukas, L., Vandewalle, J.: Weighted least squares support vector machines: Robustness and sparse approximation. Neurocomputing **48**, 85–105 (2002)
20. Suykens, J., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters **9**, 293–300 (1999)
21. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B **58**, 267–288 (1996)
22. Tong, S., Chang, E.: Support vector machine active learning for image retrieval. Proceedings of the Ninth Association for Computing Machinery International Conference on Multimedia pp. 107–118
23. Vapnik, V.: An overview of statistical learning theory. IEEE Transactions on Neural Networks **10**, 988–999 (1999)
24. Wang, H., Li, R., Tsai, C.: Tuning parameter selectors for the smoothly clipped absolute deviation method. Biometrika **94**, 553–568 (2007)