Artificial Intelligence in Medicine (2005) **xxx**, xxx—xxx

ARTIFICIAL INTELLIGENCE IN MEDICINE

http://www.intl.elsevierhealth.com/journals/aiim

Granular support vector machines with association rules mining for protein homology prediction

Yuchun Tang[*], **Bo Jin**, **Yan-Qing Zhang**

Department of Computer Science, Georgia State University, P.O. Box 3994, Atlanta, GA 30302, USA

Received 30 October 2004; received in revised form 14 January 2005; accepted 22 February 2005

KEYWORDS
Protein homology prediction;
Binary classification;
Granular computing;
Granular support vector machines;
Association rules

Summary

Objective: Protein homology prediction between protein sequences is one of critical problems in computational biology. Such a complex classification problem is common in medical or biological information processing applications. How to build a model with superior generalization capability from training samples is an essential issue for mining knowledge to accurately predict/classify unseen new samples and to effectively support human experts to make correct decisions.
Methodology: A new learning model called granular support vector machines (GSVM) is proposed based on our previous work. GSVM systematically and formally combines the principles from statistical learning theory and granular computing theory and thus provides an interesting new mechanism to address complex classification problems. It works by building a sequence of information granules and then building support vector machines (SVM) in some of these information granules on demand. A good granulation method to find suitable granules is crucial for modeling a GSVM with good performance. In this paper, we also propose an association rules-based granulation method. For the granules induced by association rules with high enough confidence and significant support, we leave them as they are because of their high "purity" and significant effect on simplifying the classification task. For every other granule, a SVM is modeled to discriminate the corresponding data. In this way, a complex classification problem is divided into multiple smaller problems so that the learning task is simplified.
Results and conclusions: The proposed algorithm, here named GSVM-AR, is compared with SVM by KDDCUP04 protein homology prediction data. The experimental results show that finding the splitting hyperplane is not a trivial task (we should be careful to select the association rules to avoid overfitting) and GSVM-AR does show significant improvement compared to building one single SVM in the whole feature space.

[*] Corresponding author. Tel.: +1 404 651 0682; fax: +1 404 463 9912.
E-mail address: ytang@gsu.edu (Y. Tang).

doi:10.1016/j.artmed.2005.02.003

Another advantage is that the utility of GSVM-AR is very good because it is easy to be implemented. More importantly and more interestingly, GSVM provides a new mechanism to address complex classification problems.

1. Introduction

Protein homology prediction between protein sequences is one of critical problems in computational biology. Protein sequences are very difficult to understand and model due to their complex random length nature. The sequential similarity measurement is believed to be useful to predict the structural or functional similarity of proteins and thus it is helpful to group proteins with similar function together. Due to this reason, it is a hot research topic for computational biologists and computer scientists in recent years. Various algorithms have been developed to measure the sequential similarity between two proteins [1,2]. From the viewpoint of data mining, protein homology prediction could be viewed as a predictive data mining task [3] because the goal is to predict the unknown value of a variable of interest given known values of other variables. More specifically, it could be modeled as a binary classification problem. If a protein sequence is homologous to a pre-specified protein sequence, it is classified to be a positive case and 1 is output, otherwise it is negative and -1 is output.

1.1. Binary data classification problems

The formal definition of a general binary classification problem is given in [4]. Notice that we assume there are l training samples, d features, and a classifier is decided by a function $f(x,\theta)$. The performance of the classifier is usually measured in terms of the sum of classification error on unseen "testing data" which is defined in Eq. (1):

$$E(y,\ f(x,\theta)) = \begin{cases} 0, & \text{if} y = f(x,\theta), \\ 1, & \text{otherwise}. \end{cases} \quad (1)$$

1.2. Binary ranking problems

Protein homology prediction could also be modeled as a binary ranking problem. In some sense, it is even more natural than a binary classification modeling. Because of the biological complexity, it is difficult and arbitrary to say two protein sequences are absolutely homologous or not (1 or -1 is output); an output with "confidence" may be more helpful. In this way, many protein sequences could be ranked by their confidence to be homologous to the pre-specified protein sequence. As a result, biologists could quickly prioritize a list of protein sequences for further study and thus their working efficiencies could be enhanced.

A binary ranking problem is similar to a binary classification problem. The differences are

- the output is a real number in the field of $[-1,1]$,
- the absolute value of the output is useless. Intuitively, a good model should rank the unseen homologous protein sequences close to the top and rank unseen non-homologous ones close to the bottom of the list.

1.3. Support vector machines

SVM is a superior classifier in that SVM embodies the structural risk minimization (SRM) principle to minimize an upper bound on the expected risk [5–9]:

$$R(\alpha) \le R_{\text{emp}}(\alpha) + \sqrt{\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l}} \quad (2)$$

where

$$R_{\text{emp}}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i,\alpha)| \text{ is empirical risk} \quad (3)$$

h is non-negative integer called the Vapnik Chervonenkis (VC) dimension, $\eta \in [0,1]$ and the bound holds with probability $1-\eta$. α is the vector of unknown parameters.

Because structural risk is a reasonable trade-off between the error on the training dataset (the first factor of Eq. (2)) and the complication of modeling (the second factor of Eq. (2)), SVM has a great ability to avoid overfitting and thus could be confidently generalized to predict new data that are not included in the training dataset.

Geometrically, SVM modeling algorithm works for a binary classification problem by constructing a separating hyperplane with maximal margin as showed in Fig. 2. Finding the optimal separating hyperplane of SVM requires the solution to a convex quadratic programming problem, the Wolfe dual formulation of which is showed in Eqs. (4)—(6) [5]:

- maximize

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (4)$$

- subject to

$$0 \leq \alpha_i \leq C, \quad (5)$$

$$\sum_i \alpha_i y_i = 0 \quad (6)$$

The geometry explanation is that the margin between classes could be maximized by maximizing L_D in Eq. (4). For linear SVM, the margin width can be calculated by Eqs. (7) and (8):

$$w = \sum_{i=1}^{N_s} \alpha_i y_i x_i \quad (7)$$

$$\text{margin width} = \frac{2}{w} \quad (8)$$

where N_s is the number of support vectors.

Kernel functions are known to be a kind of elegant dimension-increasing-based methods [5]. Nonlinear kernel functions are introduced to implicitly map input sample from input feature space into a higher dimensional feature space, where a linear classification decision could be made. Some most common nonlinear kernel functions are listed in [5].

However, a SVM halves the whole feature space (whatever original input feature space or after-kernel-transformed feature space) by a single contiguous hyperplane. It looks more or less arbitrary. What if we split the whole feature space into a set of subspaces and then build a SVM for each mixed subspace? (Here "mixed" means that the data of both classes present in the subspace. That is, the subspace is not pure).

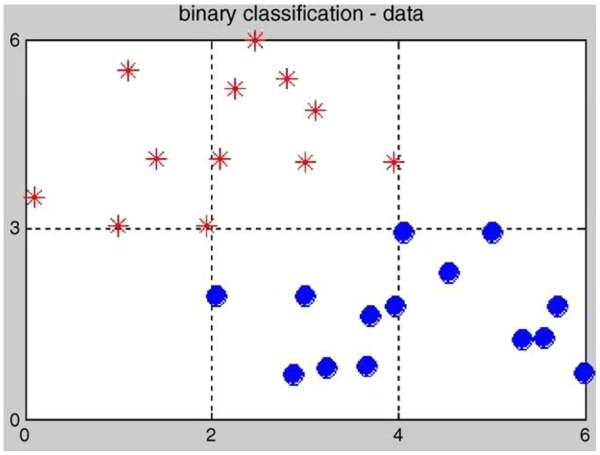


Figure 1 An artificial linear separable two-dimension binary classification problem.

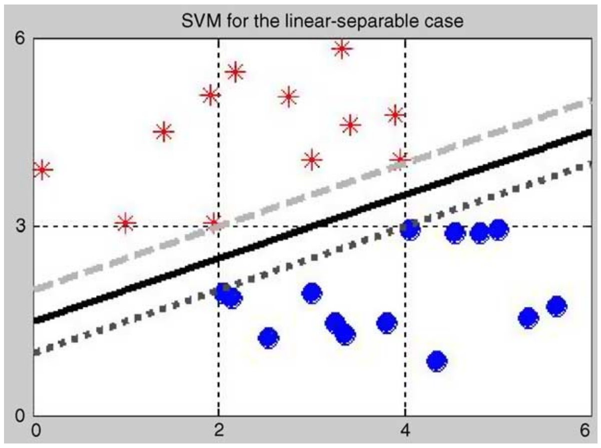


Figure 2 SVM with maximal margin.

Figs. 1—3 show a simple artificial two-dimension example: if we split the whole space by $x = 2$ and 4, and then build a SVM for subspace $2 \leq x < 4$ (because the purity of other two subspaces, we do not need to build a SVM classifier in them), obviously larger margins could be achieved and thus better generalization capability could be achieved.

The simple observation above motivates us to design a new model named GSVM on the shine of granular computing.

1.4. Granular computing

Granular computing represents information in the form of some aggregates (called "information granules") such as subsets, classes, and clusters of a universe and then solves the targeted problem

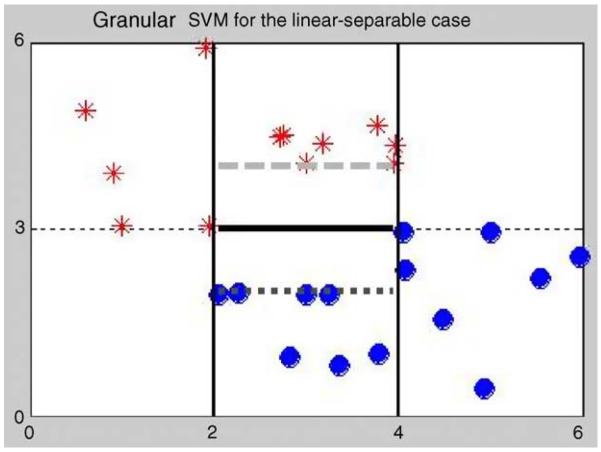


Figure 3 GSVM split the whole feature space with $x = 2$ and 4. A SVM is built for granule $2 \leq x < 4$ (named 'mixed granule'). For other two pure granules (named 'negative pure granule' and 'positive pure granule', respectively), there is no need to build classifiers. As a result, the larger margin is achieved and thus better generalization capability is expected.

in each information granule [10—12]. On one hand, for a huge and complicated task, it embodies Divide-and-Conquer principle to split the original problem into a sequence of more manageable and smaller subtasks. On the other side, for a sequence of similar little tasks, it comprehends the problem at hand without getting buried in all unnecessary details. As opposed to traditional data-oriented numeric computing, granular computing is knowledge-oriented [12]. From the data mining viewpoint, if built reasonably, information granules can make the mining algorithms more effective and at the same time avoid the notorious overfitting problem. Some formal models of information granules are:

- set theory and interval analysis,
- fuzzy sets,
- rough sets,
- probabilistic sets,
- decision trees,
- clusters,
- association rules,

1.5. Association rules

Many previous works have reported that the frequent patterns occurred in the training dataset of a complex and huge classification problem could lead to measured improvement on testing accuracy [13,14]. The idea was named ''association classification'' [14].

For a binary classification problem with continuous features, an association rule is usually formed as

$$\text{if} a_1 \in [v_{11}, v_{12}] \text{and} a_2 \in [v_{21}, v_{22}] \text{ and} \ldots a_n \in [v_{n1}, v_{n2}], \text{then} y = 1 (\text{or} -1) \quad (9)$$

The support and confidence of an association rule for a binary classification problem are defined in Eqs. (10) and (11):

$$\text{SUP(AR)} = \frac{S_{\text{PG}}}{S_{\text{W}}} \quad (10)$$

$$\text{COF(AR)} = \frac{S_{\text{PG}}}{S_{\text{G}}} \quad (11)$$

where S_{W} is the size of training data with the same class label as the THEN-part of the association rule, S_{G} the size of training data that satisfy the IF-part, while S_{PG} is the size of training data correctly classified by the association rule. Notice that S_{W} is defined in such a way that the support and confidence of an association rule are calculated based on a single class. As a result, the association rule mining will not be biased for major class in an unbalanced binary classification problem.

From Eq. (9), an association rule (or a set of association rules combined disjunctively) could be used to partition the feature space to find an information granule. So association rules mining is a possible solution for granulation. The realization of a successful ''association granulation'' depends on the following two issues.

An association rule with high enough confidence could deduce a ''pure'' granule, in which it is unnecessary to build a classifier because of its high purity. If its support is also high, it could significantly simplify and speed up classification because it decreases the size of the training dataset.

A more general association rule with a shorter IF-part should be more possible to avoid overfitting training dataset. A short IF-part means a low model complication, which in turn means a good generalization possibility.

1.6. Related works and comments

Zhang's Granular Neural Networks [15] introduced how to use neural networks to discover granular knowledge and how to use discovered granular knowledge to predict missing data. It is a combination of neural network and granular computation. There are also many works that combine SVM (or other statistical learning theory based models) with some granular computing technologies. Bennett's Support Vector Decision Trees [16] tried to create an optimal decision tree by applying the ideas from SVM, and Yu's Clustering-based SVM [17] tried to take advantage of the merit of data clustering to get high scalability for SVM while also generate the high classification accuracy.

There were already some works to take advantages of association rules mining to help to get a more accurate classifier. Yin and Han proposed a classification approach, named CPAR, which combines the advantages of both associative classification and traditional rule-based classification such as many decision trees approaches [14]. An interesting paper by She et al. proposed to apply a SVM-based method for a computational biology problem: outer membrane proteins prediction. They used subsequences that occur frequently in outer membrane sequences as input feature space for SVM modeling [13].

However, to the best of our knowledge, up to now nobody has tried to combine these two fields formally and systematically for modeling data mining problems, although each of them has already be widely applied to data mining problems and get promising results separately. GSVM tried to fill in the gap [4].

The rest of the paper is organized as follows. In Section 2, the framework of GSVM is detailed and enriched based on our previous work [4]; then an algorithm named GSVM-AR for modeling a GSVM is proposed. In Section 3, KDDCUP04 protein homology prediction task is used to show the superiority of GSVM-AR compared to building a general SVM in the whole feature space for linear kernel and RBF kernel on four performance metrics. Section 4 concludes the paper and also directs the future works.

2. Granular support vector machines

2.1. Framework and general ideas

SVM is inherently a contiguous model in that it uses a single contiguous hyperplane to halve the whole feature space. Is it reasonable to always assume that the classification boundary is contiguous? Here, we argue that the boundary maybe discrete for many classification problems. So if we can somehow correctly split the whole feature space into a set of subspaces (information granules) and then build a SVM for some mixed ones of the subspaces, the resulting model is expected to capture the inherent data distribution of the classification problem at hand more accurately. Even for a contiguous classification boundary, the boundaries from suitably built subspaces could approximate it with enough accuracy. Currently, for the discrete or other linear non-separable classification problems, the only method is to use some kernel function to map the data to a new feature space in which the data is expected to be linear separable. But up to now no kernel function can guarantee the "linear separability".

A new learning model named GSVM is proposed based on our previous work [4]. The framework of GSVM is detailed and enriched here. GSVM is a model, which systematically and formally combines the principles from statistical learning theory and granular computing theory. It works by building a sequence of information granules and then building SVM in some of information granules on demand.

Some potential advantages of GSVM are

- GSVM can get better generalization in a linear separable classification problem. Fig. 3 already shows that GSVM may improve the generalization capability by enlarging the margin width.
- GSVM can increase a linear non-separable problem's "linear separability", or even transfer a linear non-separable problem to totally linear separable. Fig. 4 shows a linear non-separable example, but if we split the whole feature space by $x = 0$ and $y = 0$, the resulting four subproblems are linear separable. Figs. 5—7 show that GSVM works for the well-known XOR problem. That

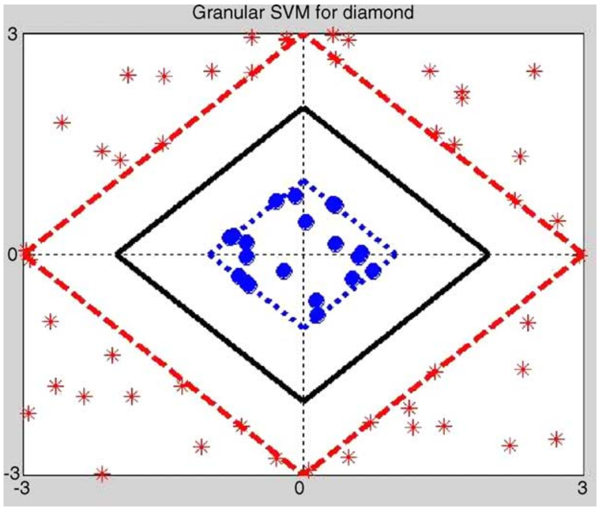


Figure 4 GSVM can transfer a linear non-separable problem to four linear separable problems by $x = 0$ and $y = 0$.

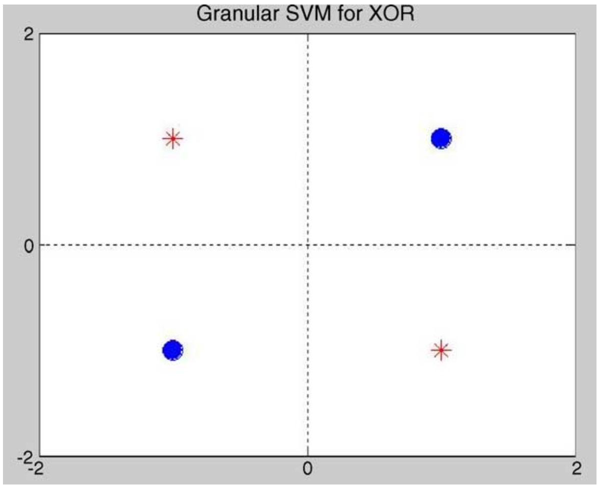


Figure 5 XOR classification problem.

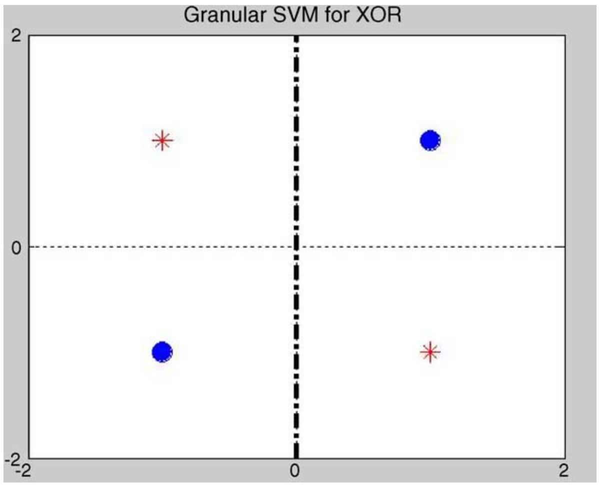


Figure 6 GSVM can transfer the well-known XOR problem to two linear separable problems by $x = 0$.

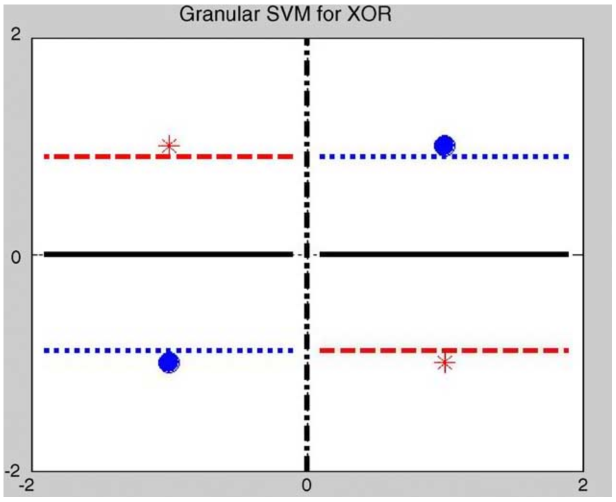


Figure 7 GSVM modeling result.

means GSVM could be a potential alternative to kernel functions by transferring a linear non-separable classification problem to a set of linear separable subproblems. In fact, these two methods are not contradictory so we can combine granule functions and kernel functions in a GSVM to achieve better separability. One way is mapping the data to a new feature space with some kernel function at first, and then a GSVM is modeled in the new feature space; another way is splitting the original feature space into a set of information granules at first, and then using different kernel functions to map the data in these information granules to different new feature spaces separately.

- In many real world data mining applications, what people expect is not only to get a model with small prediction error, but also to explain the reason why it works so well. As we know, SVM is almost unable to provide this kind of information. Patterns and rules, if used as granular functions, should grant a GSVM the ability to be easier to explain.
- Compared to the original SVM modeling method of optimizing the parameters by cross-validation, the GSVM modeling algorithm searches the splitting hyperplane by extending maximal margin principle. As a result, GSVM runs faster, more robust to noise data, and more stable to SVM parameters.
- GSVM is easier to be parallelized so that the time for modeling could be decreased even further. As a result, GSVM is more appropriate to be applied to huge data classification problems.
- Like SVM, GSVM could also be applied to multiple classification or regression problems without or with small modifications.

However, building suitable information granules is far from a trivial task. The key is to build the information granules somehow reasonably and effectively. Many questions need to be answered during modeling a GSVM:

- Top-down or bottom-up? Begin from the whole feature space and then gradually split it into smaller spaces (top-down)? Or begin from creating many tiny information granules and then gradually combine them into larger spaces?
- What is the stop condition? What time should we stop splitting or combining? How do we know a splitting or combining will result in overfitting? From the granular computing viewpoint, how do we know we already get optimal information granules? Notice here maybe the original whole feature space is itself an optimal granule so we even do not need to split it.
- If top-down, how to find the feature(s) the splitting hyperplane should be based on? Is it reasonable to split a space by a hyperplane orthogonal to a single feature? Or select a splitting hyperplane based on a group of features?
- After selecting the splitting feature (or features), how to decide the direction and the bias parameters of the splitting hyperplane $wx + b$?

2.2. A GSVM modeling method by association rules

This paper proposes to take advantage of association rules mining for modeling a GSVM in the top-down way. The hyperplane used to split the feature space is selected according to mined association rules with high confidence and significant support. Confidence of a good association rule should be as high as possible (should be at least higher than the validation accuracy of the best SVM in the whole space), while its support cannot be too small, otherwise it is not useful (in other words, the support should be significant). Fig. 8 describes the GSVM-AR modeling algorithm. The basic idea is to extend PPG and NPG iteratively until GSVM-AR gets the best validation performance. If necessary, the cross-validation method could be used. Notice the support threshold is provided as an input, and the confidence threshold is set to be the validation accuracy of the general SVM in the whole feature space. For each feature, at most two association rules are mined. Therefore, if the time complexity for modeling a general SVM is $O(l^2d)$, the time complexity for modeling a GSVM is $O(l^2d^2)$ and dominated by the *while* loop to find a GSVM with the best

```
MiningOneFeatureARs(TrainingData T, SupportThreshold Sth, ConfidenceThreshold Cth)
{
   y = the class label vector in T;
   WAR = empty set;

   for each input feature x
   {
      PR = {r | r is an association rule with the format "if x0<x<x1,then y=1", support>=Sth, and confidence >=Cth in T};
      R = {r | r is the rule with the highest confidence in PR};
      WAR = WAR + {r | r is the rule with the highest support in R};
      NR = {r | r is an association rule with the format "if x0<x<x1,then y=-1", support>=Sth, and confidence >=Cth in T};
      R = {r | r is the rule with the highest confidence in NR};
      WAR = WAR + {r | r is the rule with the highest support in R};
   }
    return WAR;
}

GSVM-AR(TrainingData T, SupportThreshold Sth)
{
   MG = WFS = the whole feature space on T;
   PPG = NPG = empty set;
   MG_SVM = the SVM modeled on the training data in MG optimized by grid search heuristic;
   GSVM-AR = {      if a sample x in PPG, then its class label y = 1;
                    if a sample x in NPG, then its class label y = -1;
                    if a sample x in MG,  then its class label y=the class label predicted by MG_SVM;
                 }
  VP = the cross validation performance of GSVM-AR in WFS;
   Cth = the cross validation accuracy of GSVM-AR in WFS;

   WAR = MiningOneFeatureARs(T,Sth,Cth);
   while(WAR is not empty)
   {
            r = the association rule in WAR such that if r is added into PPG or NPG,
                 the purity of PPG or NPG is the highest compared to adding any other rule in WAR;
            WAR = WAR - {r};

            if r is a positive rule
            {
                     newPPG = PPG + {r};
                     newNPG = NPG;
            }
            else
            {
                     newPPG = PPG;
                     newNPG = NPG + {r};
            }
            newMG = WFS - newPPG - newNPG;
            newMG_SVM = the SVM modeled on the training data in newMG optimized by grid search heuristic;
            newGSVM-AR = {    if a sample x in newPPG, then its class label y = 1;
                              if a sample x in newNPG, then its class label y = -1;
                              if a sample x in newMG,  then its class label y=the classlabel predicted by newMG_SVM;
                           }
            newVP = the cross validation performance of newGSVM-AR in WFS;
            if newVP is better than VP
            {
                     PPG = newPPG;
                     NPG = newNPG;
                     MG = newMG;
                     GSVM-AR = newGSVM-AR;
                     VP = newVP;
            }
   }
   return GSVM-AR;
}
```

Figure 8 GSVM-AR modeling algorithm by C-style pseudocode.

validation performance. Notice the time complexity of *MiningOneFeatureARs* is $O(ld)$.

Decision tree (DT) is a popular model for granulation and classification. Many works have been proposed to combine DT with SVM. For example, Bennett's support vector decision trees [16] tried to create an optimal decision tree by applying the ideas from SVM. However, we notice that many other models, such as association rules, fuzzy sets, or clustering, can also be utilized to find suitable granules. In this sense, GSVM is a more general framework than DT-SVM. On the other hand, GSVM-AR utilizes simple association rules for granulation and thus is easier to be implemented compared with DT-SVM.

3. Experimental evaluation

The performance of the GSVM-AR modeling algorithm proposed in this paper is compared with building one single SVM in the whole feature space. We make comparisons on linear SVM versus linear GSVM-AR and RBF kernel SVM versus RBF kernel GSVM-AR. The hardware we used is a PC with P4-2.8 MHz CPU and 256 M RAM. The software we developed is based on OSU SVM Classifier Matlab Toolbox [18] which implements a Matlab interface to LIBSVM [19].

3.1. Data description

KDDCUP04 protein homology prediction task [20] is used for experiment. The detailed characteristics of the dataset are listed in Table 1. From the table, we can see that the task could be modeled as a binary classification or a binary ranking problem: given a protein sequence, the task is to predict whether it is homologous to the corresponding native sequence or not. There are 153 native sequences in the training dataset and 150 native sequences in the testing dataset. For each native sequence, there is a block of approximately 1000 protein sequences with class label (1 means homologous and 0 means non-homologous). The class labels of protein sequences in testing dataset are unknown. Seventy-four features are provided to describe the match (e.g. the score of a sequence alignment) between the native protein sequence and the sequence that is tested for homology. We can also see that the problem is highly unbalanced: there are only 1296 homologous protein sequences from altogether 145751 ones in the training dataset.

Table 1 Characteristics of Kddcup04 protein homology prediction datasets

Dataset	Block	Size	Attr	Class	Ratio
Training	153	≈1000	74	2	1296/144455
Testing	150	≈1000	74	2	N/A

Block = # of blocks, Size = # of protein sequences in each block, Attr = # of input features, Class = # of classes, Ratio = # of homologous sequences/# of non-homologous sequences. The data is without missing data.

Four metrics are used for performance measures:

- TOP1: fraction of blocks with a homologous sequence ranked top 1 (maximize),
- RKL: average rank of the lowest ranked homologous sequence (minimize),
- RMS: root mean squared error averaged on blocks (minimize),
- APR: average of the average precision in each block. For a single block, APR could be approximately described as the area of precision-recall curves (maximize).

RMS is a metric for accuracy evaluation, but is easier to show the differences between models than directly using error values. The other three metrics are rank-based, which means that the three metrics' values are decided by the order of ranking list, and the absolute values of predictions do not affect the performances. The four metrics are precisely defined in perf [21]. In our experiment, we use the corresponding code to calculate the four metrics.

Because of the absent of the class label in the testing dataset, only the training dataset is used in our experiment. And because our result is not on the original testing dataset, so it could not be compared with the current best results on the competition. That is, our goal is not to be involved in the competition to get the best result, but to use the data to show GSVM-AR's superiority to SVM.

3.2. Data preprocessing

Firstly, we scale and normalize the input features to [−0.9, 0.9]. The scaling is on each different block separately. The reason is that the protein sequences in different blocks are in different protein families, which are so remote that the similar absolute feature vectors cannot mean similar homology behaviors. However, to avoid overfitting, the association rules are mined from non-scaled original data.

After scaling, we make five trials. In each trial, the data is randomly split into training dataset and testing dataset with the conditions in Eqs. (12)–(14). That is, 102 blocks are used for training and other 51 blocks used for testing:

$$S(\text{training}) : S(\text{testing}) = 2 : 1 \tag{12}$$

$$S(\text{positive\_training}) : S(\text{positive\_testing}) = 2 : 1 \tag{13}$$

$$S(\text{negative\_training}) : S(\text{negative\_testing}) = 2 : 1 \tag{14}$$

$S(x)$ means the number of blocks in the dataset x.

3.3. Modeling

In each trial, we select just 1 block for modeling and other 101 training blocks for validation. That is because our preliminary tests also show that it is even worse if we mix multiple blocks together for training a model. (We also skip the details because it is out of the scope of this paper.)

Two models are created for performance comparison. The first one is a general SVM in the whole space. The parameters of the SVM are optimized by grid search heuristic [22].

In linear SVM, the regulation parameter C is optimized by grid search heuristic at Eq. (15):

$$C \in \{2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}, 2^{-7.5}, 2^{-7}, 2^{-6.5}, 2^{-6}, 2^{-5.5}, 2^{-5}, 2^{-4.5}, 2^{-4}, 2^{-3.5}, 2^{-3}, 2^{-2.5}, 2^{-2}, 2^{-1.5}, 2^{-1}, 2^{-0.5}, 2^{0}, 2^{0.5}, 2^{1}, 2^{1.5}, 2^{2}\} \quad (15)$$

The RBF kernel parameters (γ, C) are optimized by grid search heuristic at Eqs. (16) and (17):

$$\gamma \in \{2^{-26}, 2^{-24}, 2^{-22}, 2^{-20}, 2^{-18}, 2^{-16}, 2^{-14}, 2^{-12}, 2^{-10}, 2^{-8}\} \quad (16)$$

$$C \in \{2^{5}, 2^{6}, 2^{7}, 2^{8}, 2^{9}, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}\} \quad (17)$$

We repeat this modeling process for each of 102 training blocks. After that, five blocks with best validation performance on a special metric are selected to build GSVM for comparison.

For GSVM-AR modeling, we mine association rules first. To avoid overfitting, the association rules should be as simple as possible. Due to this reason, only 1-feature association rules with the format $x_0 \le x < x_1$ is mined. And only the rules with confidence higher than the general SVM's validation accuracy and significant support are kept as candidates.

The mined association rules with their support and confidence are listed in Table 2. In the table, the support and confidence for each rule are listed. For example, the sixth row shows a positive association rule. In trial 1, the training dataset has 879 homologous protein sequences, 547 ones of which satisfy IF-part of the rule, and 543 ones satisfy both IF-part and THEN-part:

$$\text{If attr58} > 8.3, \text{ then} y = 1 \text{ with}$$
$$\text{Confidence} = 543/547 = 99.27\%,$$
$$\text{Support} = 543/879 = 61.77\%.$$

The second row shows a negative association rule. In trial 1, the training dataset has 11642 protein sequences satisfy IF-part of the rule. And all of them satisfy THEN-part too:

$$\text{If attr45} > -4, \text{ then} y = -1 \text{ with}$$
$$\text{Confidence} = 11642/11642 = 100\%,$$
$$\text{Support} = 11642/96055 = 12.12\%.$$

After that, we iteratively combine association rules by disjunction to find the granules that are both pure and significant. When the process is completed, three granules are created: the granule induced by negative rules is named NPG because almost all protein sequences in the granule are non-homologous; the granule induced by positive rules is named PPG due to the similar reason; and the remaining space is named "Mixed Granule" (MG), in which a SVM with the same kernel as the general SVM is built. The three granules are decided by Eqs. (18)–(20):

$$\text{PPG} = \cup \text{positive association rules}, \quad (18)$$

$$\text{NPG} = \cup \text{negative association rules} - \text{PPG}, \quad (19)$$

$$\text{MG} = \text{WFS} - \text{PPG} - \text{NPG}, \quad (20)$$

where WFS means the whole feature space.

Notice that the overlapping area of PPG and NPG is accounted in PPG. That means the granulation is biased for homologous proteins to compensate for its minority.

For the protein prediction task:

- PPG is formed by

  $\text{If attr58} > 8.3, \text{ then} y = 1$

Table 2 1-Feature association rules on original unscaled training data with confidence/support in five trials

Rule	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
If attr5 > 78, then $y = -1$	100%/8666	100%/8976	100%/9141	100%/8992	100%/8753
If attr45 > −4, then $y = -1$	100%/11642	100%/11703	100%/11075	100%/11234	100%/11786
If attr53 < −2.06, then $y = -1$	100%/1080	100%/1041	100%/1060	100%/987	100%/1029
If attr58 < −1.53, then $y = -1$	100%/1714	100%/1727	100%/1758	100%/1712	100%/1711
If attr68 < −2.21, then $y = -1$	100%/2515	100%/2608	100%/2610	100%/2404	100%/2725
If attr58 > 8.3, then $y = 1$	99.27%/547	99.56%/457	99.35%/465	99.42%/517	99.41%/510

Trial 1 (positive:negative = 879:96055 in the training dataset), trial 2 (positive:negative = 909:95911 in the training dataset), trial 3 (positive:negative = 810:96139 in the training dataset), trial 4 (positive:negative = 886:95580 in the training dataset), trial 5 (positive:negative = 919:96892 in the training dataset).

Table 3 TOP1 on validation/test set in five trials (mean ± S.D. from best five blocks)

Trial	Validation data with linear kernel (%)	Testing data with linear kernel (%)	Validation data with RBF kernel (%)	Testing data with RBF kernel (%)
1	91.09 ± 0.00/91.09 ± 0.00	78.82 ± 2.56/79.61 ± 2.23	91.09 ± 0.70/91.09 ± 0.70	78.82 ± 2.56/79.61 ± 2.23
2	85.55 ± 0.54/85.55 ± 0.54	90.98 ± 1.07/90.98 ± 1.07	85.55 ± 0.54/85.55 ± 0.54	90.59 ± 1.64/90.59 ± 1.64
3	86.14 ± 0.00/86.14 ± 0.00	89.42 ± 1.75/89.42 ± 1.75	86.14 ± 0.70/86.14 ± 0.70	89.02 ± 1.07/89.02 ± 1.07
4	87.72 ± 0.54/87.72 ± 0.54	85.88 ± 3.22/85.88 ± 3.22	87.53 ± 0.54/87.53 ± 0.54	85.49 ± 4.07/85.88 ± 3.22
5	90.50 ± 0.54/90.69 ± 0.54	81.57 ± 1.07/81.57 ± 1.07	90.30 ± 0.44/90.50 ± 0.54	81.57 ± 1.07/81.96 ± 0.88

Best five blocks in trial 1 are 210, 103, 73, 69, and 16, best five blocks in trial 2 are 170, 65, 274, 255, and 236, best five blocks in trial 3 are 210, 162, 144, 65, and 64, Best five blocks in trial 4 are 170, 65, 16, 289, and 274, best five blocks in trial 5 are 73, 16, 261, 255, and 252.

Table 4 RKL on validation/test set in five trials (mean ± S.D. from best five blocks)

Trial	Validation data with linear kernel	Testing data with linear kernel	Validation data with RBF kernel	Testing data with RBF kernel
1	67.69 ± 2.38/62.63 ± 1.48	93.62 ± 17.23/85.00 ± 14.44	66.88 ± 2.48/61.79 ± 1.61	92.21 ± 16.92/83.67 ± 13.93
2	91.33 ± 6.01/83.73 ± 4.45	41.29 ± 8.46/35.94 ± 7.35	87.60 ± 5.97/81.16 ± 5.08	35.41 ± 7.48/31.06 ± 5.85
3	71.39 ± 2.99/64.80 ± 3.33	88.05 ± 8.97/81.84 ± 7.80	66.34 ± 5.15/61.17 ± 5.05	90.99 ± 6.77/84.51 ± 5.66
4	73.53 ± 6.81/68.05 ± 5.82	80.02 ± 10.45/73.42 ± 8.59	72.42 ± 6.15/67.27 ± 5.33	78.11 ± 8.61/71.91 ± 7.40
5	71.64 ± 1.71/65.60 ± 2.31	87.11 ± 6.97/78.83 ± 5.55	68.91 ± 4.20/63.70 ± 4.02	82.29 ± 8.57/75.11 ± 6.96

Best five blocks in trial 1 are 55, 164, 303, 135, and 266, best five blocks in trial 2 are 55, 110, 13, 69, and 73, best five blocks in trial 3 are 289, 110, 2, 164, and 69, best five blocks in trial 4 are 55, 289, 164, 25, and 65, best five blocks in trial 5 are 13, 110, 73, 164, and 16.

Table 5 RMS on validation/test set in five trials (mean ± S.D. from best five blocks)

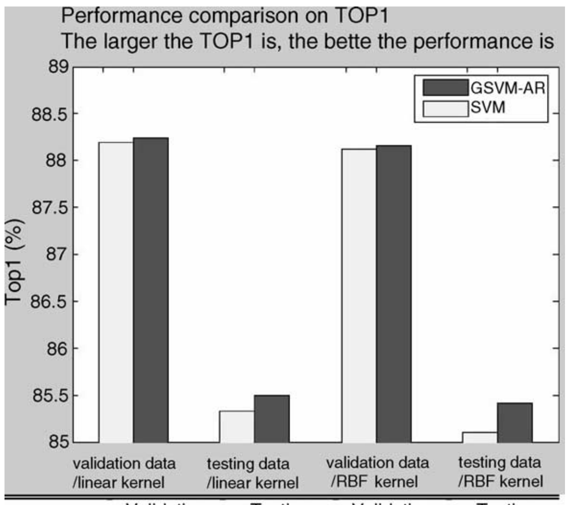
Trial	Validation data with linear kernel (%)	Testing data with linear kernel (%)	Validation data with RBF kernel (%)	Testing data with RBF kernel (%)
1	5.33 ± 0.03/3.87 ± 0.21	5.90 ± 0.09/5.24 ± 0.20	5.31 ± 0.03/3.87 ± 0.23	5.88 ± 0.06/5.25 ± 0.20
2	5.79 ± 0.04/4.66 ± 0.09	4.98 ± 0.14/3.46 ± 0.09	5.79 ± 0.04/4.66 ± 0.09	4.98 ± 0.14/3.46 ± 0.09
3	5.32 ± 0.06/4.05 ± 0.09	5.95 ± 0.07/4.63 ± 0.08	5.31 ± 0.05/4.05 ± 0.09	5.95 ± 0.07/4.63 ± 0.07
4	5.49 ± 0.05/4.19 ± 0.06	5.47 ± 0.09/4.49 ± 0.08	5.49 ± 0.05/4.13 ± 0.02	5.45 ± 0.09/4.44 ± 0.11
5	5.57 ± 0.02/4.22 ± 0.14	5.40 ± 0.07/4.20 ± 0.02	5.62 ± 0.12/4.21 ± 0.15	5.45 ± 0.09/4.21 ± 0.04

Best five blocks in trial 1 are 256, 103, 277, 271, and 212, best five blocks in trial 2 are 73, 48, 238, 256, and 277, best five blocks in trial 3 are 103, 212, 60, 7, and 48, best five blocks in trial 4 are 256, 231, 73, 277, and 103, best five blocks in trial 5 are 60, 16, 103, 7, and 48.

Table 6 APR on validation/test set in five trials (mean ± S.D. from best five blocks)

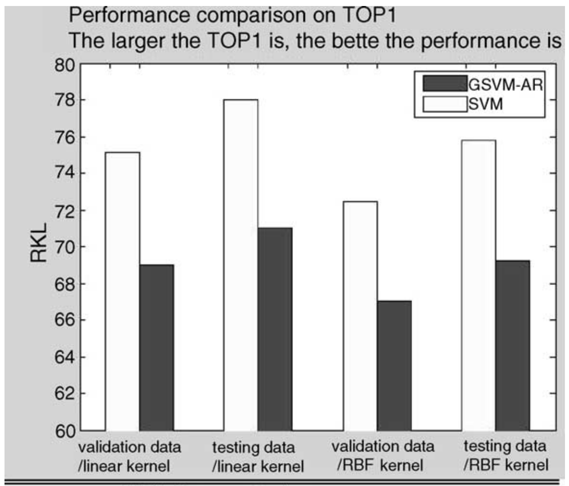
Trial	Validation data with Linear kernel (%)	Testing data with Linear kernel (%)	Validation data with RBF kernel (%)	Testing data with RBF kernel (%)
1	83.30 ± 0.47/83.53 ± 0.57	75.78 ± 1.23/76.38 ± 0.86	83.28 ± 0.52/83.49 ± 0.60	75.94 ± 1.37/76.51 ± 0.96
2	77.61 ± 0.24/77.77 ± 0.21	86.19 ± 1.22/86.47 ± 1.20	77.52 ± 0.26/77.77 ± 0.15	86.06 ± 1.12/86.34 ± 1.13
3	78.39 ± 0.86/78.68 ± 0.87	82.92 ± 0.84/83.01 ± 0.85	78.36 ± 0.88/78.60 ± 0.89	83.08 ± 1.08/83.14 ± 1.09
4	81.45 ± 0.36/81.66 ± 0.45	79.39 ± 1.36/79.72 ± 1.41	81.41 ± 0.45/81.63 ± 0.54	79.08 ± 1.78/79.38 ± 1.85
5	83.39 ± 0.44/83.63 ± 0.45	76.03 ± 0.55/76.28 ± 0.56	83.31 ± 0.65/83.56 ± 0.68	76.16 ± 0.53/76.42 ± 0.59

Best five blocks in trial 1 are 16, 27, 73, 255, and 103, best five blocks in trial 2 are 55, 73, 163, 256, and 170, best five blocks in trial 3 are 103, 64, 60, 274, and 243, best five blocks in trial 4 are 16, 27, 73, 170, and 274, best five blocks in trial 5 are 16, 73, 163, 103, and 170.



SVM/ GSVM-AR	Validation data with Linear kernel (%)	Testing data with Linear kernel (%)	Validation data with RBF kernel (%)	Testing data with RBF kernel (%)
Mean	**88.20/88.24**	**85.33/85.49**	**88.12/88.16**	**85.10/85.41**
Std	**2.32/2.36**	**5.07/4.84**	**2.32/2.37**	**5.00/4.60**

Figure 9 Performance comparison on TOP1 metric averaged on five trials. The larger TOP1 is, the better the performance is. The results are grouped by different data/kernel pairs. In each group, the left bar shows the result of SVM, while the right GSVM-AR. The mean and standard deviation statistics are given in the above table. In each cell, the first number is the result of SVM, while the second GSVM-AR.



SVM/ GSVM-AR	Validation data with Linear kernel	Testing data with Linear kernel	Validation data with RBF kernel	Testing data with RBF kernel
Mean	**75.12/68.96**	**78.02/71.01**	**72.43/67.02**	**75.80/69.25**
Std	**9.42/8.48**	**21.72/20.17**	**9.24/8.56**	**23.31/21.56**

Figure 10 Performance comparison on RKL metric averaged on five trials. The smaller RKL is, the better the performance is. The results are grouped by different data/kernel pairs. In each group, the left bar shows the result of SVM, while the right GSVM-AR. The mean and standard deviation statistics are given in the above table. In each cell, the first number is the result of SVM, while the second GSVM-AR.

- NPG is formed by

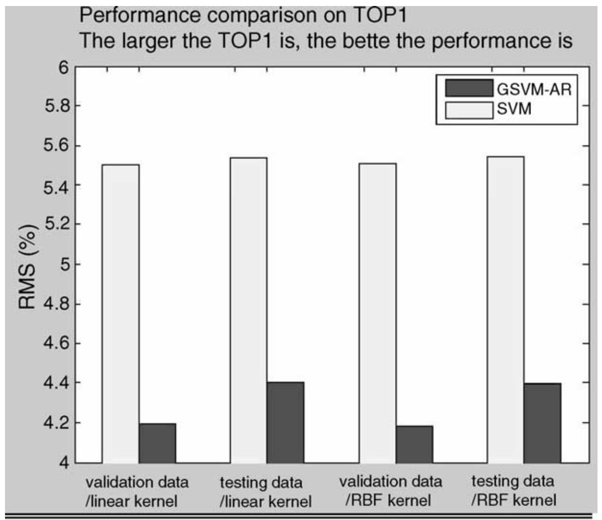
  If $\text{attr58} \le 8.3$ and($\text{attr5} > 78$ or $\text{attr45} > -4$
  or $\text{attr53} < -2.06$ or $\text{attr58} < -1.53$
  or $\text{attr68} < -2.21$), then$y = -1$

And then we compare two models on the top five blocks for the four metrics. For protein sequences in the PPG and NPG, the outputs are 1s or −1s, respectively. Because SVM is originally designed for binary classification problems, for protein sequences in MG, if a metric is rank-based, we adopt the distance from the predicted protein sequence to the separating hyperplane (normalized to be in [−1,1]) as its output.

3.4. Results

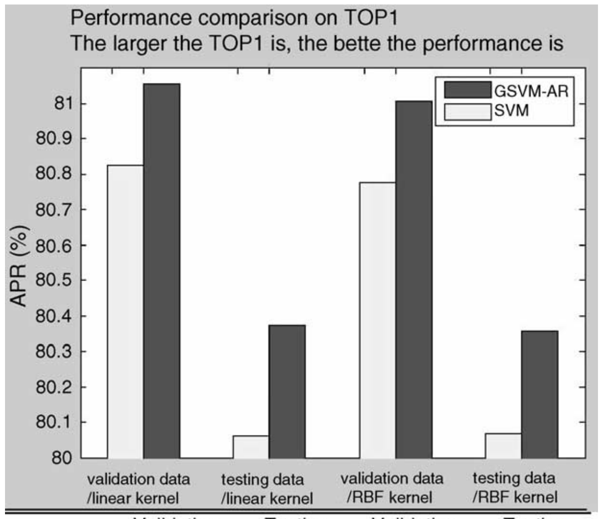
The experimental results are reported in Tables 3—6 and Figs. 9—12. In each cell of a table, the performance of SVM is reported as the first number, while the performance of GSVM-AR as the second number.

For TOP1 metric, Table 3 and Fig. 9 show that the performance of GSVM-AR is a little better than SVM with both linear kernel (from 85.33% to 85.49% for testing data) and RBF kernel (from 85.10% to 85.41%



SVM/ GSVM-AR	Validation data with Linear kernel (%)	Testing data with Linear kernel (%)	Validation data with RBF kernel (%)	Testing data with RBF kernel (%)
Mean	**5.50/4.20**	**5.54/4.41**	**5.50/4.18**	**5.54/4.40**
Std	**0.18/0.29**	**0.37/0.60**	**0.19/0.30**	**0.37/0.60**

Figure 11 Performance comparison on RMS metric averaged on five trials. The smaller RMS is, the better the performance is. The results are grouped by different data/kernel pairs. In each group, the left bar shows the result of SVM, while the right GSVM-AR. The mean and standard deviation statistics are given in the above table. In each cell, the first number is the result of SVM, while the second GSVM-AR.



SVM/ GSVM-AR	Validation data with Linear kernel (%)	Testing data with Linear kernel (%)	Validation data with RBF kernel (%)	Testing data with RBF kernel (%)
Mean	**80.83/81.06**	**80.06/80.37**	**80.78/81.01**	**80.07/80.36**
Std	**2.52/2.53**	**4.22/4.12**	**2.54/2.54**	**4.20/4.09**

Figure 12 Performance comparison on APR metric averaged on five trials. The larger APR is, the better the performance is. The results are grouped by different data/kernel pairs. In each group, the left bar shows the result of SVM, while the right GSVM-AR. The mean and standard deviation statistics are given in the above table. In each cell, the first number is the result of SVM, while the second GSVM-AR.

for testing data). For example, for testing data with linear kernel, averagely to say, there are 51 × 85.33% = 43.52 blocks with a homologous protein sequence as the TOP1 in the ranking list predicted by SVM, while 51 × 85.49% = 43.60 blocks by GSVM-AR. The improvement is small because the protein sequences ranked as TOP1 in the lists are easiest to be predicted. So a general SVM is good enough to predict them.

For RKL metric, Table 4 and Fig. 10 show that GSVM-AR significantly outperform SVM. That is, the average rank of the lowest ranked homologous sequences is decreased significantly (from 78.02% to 71.01% for testing data with linear kernel, from 75.80% to 69.25% for testing data with RBF kernel). When recall is set to be 1, GSVM-AR has higher precision than SVM. As a result, homologous sequences are clearer to be differentiated from non-homologous ones with GSVM-AR than with SVM.

For RMS metric, Table 5 and Fig. 11 show that the performance of GSVM-AR is also significantly better than SVM. That is, the average root mean squared error is decreased significantly (from 0.0554% to 0.0441% for testing data with linear kernel, from 0.0554% to 0.0440% for testing data with RBF kernel). That means GSVM-AR is more accurate. For example, approximately, for a testing block with 1000 protein sequences, 3.07 protein sequences are misclassified by SVM with RBF kernel, while only 1.94 ones are misclassified by GSVM-AR with RBF kernel.

For APR metric, Table 6 and Fig. 12 show that the performance of GSVM-AR is also better than SVM with both linear kernel (from 80.06% to 80.37%) and RBF kernel (from 80.07% to 80.36%).

The standard deviations in these tables show that experiment results are stable and conceivable.

4. Conclusions

A new learning model called Granular Support Vector Machines is proposed based on our previous work [4]. GSVM systematically and formally combine the ideas from statistical learning theory and granular computing. It works by building a sequence of information granules and then building a SVM in each of the mixed information granules.

In this paper, we also give an implementation method named GSVM-AR for modeling a GSVM by building information granules in the top-down way with the aid of association rules. GSVM-AR works by building three information granules, called Positive Pure Granule, Negative Pure Granule, and Mixed Granule, respectively. Because of being generated from association rules with high confidence and significant support, the PPG and NPG have high purity. Therefore, we only need to build a Support Vector Machine in MG.

The experimental results on KDDCUP04 protein homology prediction task show that finding the splitting hyperplane is not a trivial task (we should be careful to select the association rules to avoid overfitting) and GSVM-AR does show significant improvement compared to building one single SVM in the whole feature space. Although the association rules are limited to be 1-feature format (that means the splitting hyperplane is limited to be orthogonal to a single feature) and the number of information granules is fixed to be three, GSVM-AR shows superior generalization capability. Another advantage is that the utility of GSVM-AR is very good because it is easy to be implemented.

More importantly and more interestingly, GSVM provides a new mechanism to address complex classification problems, which are common in medical or biological information processing applications. The modeling method for a GSVM proposed here is just one step into this interesting research topic. The open problem is that how to get the optimal or suboptimal information granules effectively and efficiently. In the future, we will try more modeling methods for GSVM to improve the model's generalization capability and also understandability.

Acknowledgements

This work is supported in part by NIH under P20 GM065762. Yuchun Tang is supported by Molecular Basis for Disease (MBD) Doctoral Fellowship, Georgia State University.

References

[1] Noble WS. Support vector machine applications in computational biology. In: Schoelkopf B, Tsuda K, Vert J-P, editors. Kernel Methods in Computational Biology. MIT Press; 2004. p. 71—92.
[2] Schölkopf B, Guyon I, Weston J. Statistical Learning and Kernel Methods in Bioinformatics. In: Frasconi P, Shamir R, editors. Artificial Intelligence and Heuristic Methods in Bioinformatics 183. Amsterdam: IOS Press; 2003. p. 1—21.
[3] Hand D, Mannila H, Smyth P. Principle of Data Mining. Cambridge, London: MIT Press, 2001.
[4] Tang YC, Jin B, Sun Y, Zhang Y-Q. Granular Support Vector Machines for Medical Binary Classification Problems. In: Gary B, Fogel, editors. Proceedings of the IEEE CIBIB. Piscataway, HJ: IEEE Computational Intelligence Society; 2004. p. 73—8.
[5] Burges CJC. A tutorial on support vector machines for pattern recognition. Data Mining Knowledge Disc 1998; 2(2):121—67.
[6] Chin KK, Support vector machines applied to speech pattern classification, Master's thesis, Engineering Department, Cambridge University, 1999.
[7] Cristianini N, Shawe-Taylor J. An introduction to support vector machines and other Kernel-based learning methods. New York: Cambridge University Press, 1999.
[8] Gunn S. Support vector machines for classification and regression, ISIS technical report, Image Speech and Intelligent Systems Group. University of Southampton; 1998.
[9] Vapnik V. Statistical Learning Theory. New York: John Wiley and Sons, 1998.
[10] Bargiela A. Granular Computing: an introduction. Kluwer Academic Pub; 2002.
[11] Yao JT, Yao YY. A granular computing approach to machine learning. In: Wang Lipo, Halgamuge Saman Kmn, Yao Xin, editors. Proceedings of the FSKD'02. Singapore: Orchid Country Club; 2002. p. 732—6.
[12] Yao YY. On Modeling data mining with granular computing. In: Proceedings of the COMPSAC. Chicago, IL, USA: IEEE Computer Society; 2001. p. 638—43.
[13] She R, Chen F. Frequent-subsequence-based prediction of outer membrane proteins. In: Getoor Lise, Senator Ted EdE, Domingos Pedro, Faloutsos Christos, editors. Proceedings of the SIGKDD'03. Washington, DC, USA: ACM press; 2003. p. 436—45.
[14] Yin X, Han J. CPAR: classification based on predictive association rules. In: Barbará Daniel, Kamath Chandrika, editors. Proceedings of the SIAM International Conference on Data Mining. San Francisco, CA, USA: SIAM; 2003. p. 331—5.
[15] Zhang Y-Q, Fraser MD, Gagliano RA, Kandel A. Granular neural networks for numerical-linguistic data fusion and knowledge discovery. IEEE Trans Neural Netw 2000;11(3): 658—67.
[16] Bennett KP, Blue J, A support vector machine approach to decision trees, R.P.I math report no. 97-100, Rensselaer Polytechnic Institute, Troy, NY, 1997.
[17] Yu H, Yang J, Han J. Classifying large data sets using SVMs with hierarchical clusters. In: Getoor Lise, Senator Ted EdE, Domingos Pedro, Faloutsos Christos, editors. Proceedings of the SIGKDD'03. Washington, DC, USA: ACM press; 2003. p. 306—15.
[18] Ma J, Zhao Y, Ahalt S, OSU SVM Classifier Matlab Toolbox, Available at http://www.ece.osu.edu/~maj/osu_svm/ (last accessed: 9 April 2005).
[19] Chang C-C, Lin C-J, LIBSVM: a library for support vector machines, 2001. Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm (last accessed 9 April 2005).
[20] http://kodiak.cs.cornell.edu/kddcup/index.html (last accessed: 9 April 2005).
[21] Caruana R, The PERF Performance Evaluation Code, http://kodiak.cs.cornell.edu/kddcup/software.html (last accessed 9 April 2005).
[22] Hsu C-W, Chang C-C, Lin C-J, A practical guide to support vector classification, Available at http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (last accessed 9 April 2005).