中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-10-001

Protein-Protein Interface Prediction based on a Novel SVM Speedup

Tsu-shu Tseng, Chien-Yang Guo, Wen-Lian Hsu, and Fu Chang

January 11 2010 || Technical Report No. TR-IIS-10-001

http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2010/tr10.html

Protein-Protein Interface Prediction based on a Novel SVM Speedup

Tsu-shu Tseng, Chien-Yang Guo, Wen-Lian Hsu[1], and Fu Chang[2]
Institute of Information Science, Academia Sinica
128 Academia Road, Taipei, 115, Taiwan
{tstseng, asdguo, hsu, fchang}@iis.sinica.edu.tw

Abstract

Protein-protein interactions play a crucial role in many cellular processes. Prediction of amino acid residues that appear in interaction sites helps decipher protein functions. Since a significant number of complexes have large enough interfaces, we hypothesize that the complex formation follows the induced-fit mechanism rather than the lock-and-key mechanism. Therefore, one should be able to characterize interface regions by frequent appearances of unstructured or flexible amino acid residues in those regions. For this residue prediction problem, we designed a novel method called "tree decomposition support vector machine" (TDSVM) that can handle large samples. Previously, the sizes of protein chains used as training data were generally in the scope of hundreds, whereas TDSVM extends the number to thousands (4,064 in our case), which yields more than a million samples, represented as feature vectors. Using TDSVM to speed up the training of kernel-based support vector machines (SVMs), at a factor of nearly 300, we were able to perform numerous experiments efficiently to optimize the parameters and feature selection that would otherwise take months. As a result, we achieved prediction outcomes with substantially high scores in F_1-measure and Matthews correlation coefficient (MCC) using only protein-sequence information.

1. Introduction

Identification of protein interaction sites has a significant impact on understanding protein functions, elucidating signal transduction networks and drug design studies (Bahadur and Zacharias, 2008).

Through a process called protein-protein recognition (Colin, 2000), two protein chains (subunits) combine to form a homodimer or a heterodimer. The interface of this protein complex is formed by many contacting residue pairs. A contacting residue pair contains two residues on different subunits where one residue on one subunit is usually 6 to 8 Angstroms away from another residue on the other subunit.

Traditionally, it was believed that well-defined protein structures determine biological functions. However, recently it has been observed that many proteins, including a large fraction of eukaryotic proteins, contain regions without a well-defined structure (Fong et al., 2009). These regions are called *intrinsically disordered regions*. Intrinsic disorder has been associated with particular functions in-

[1] Corresponding author 1.
[2] Corresponding author 2.

cluding cell regulation, signaling, protein-DNA binding, and protein-ligand binding.

Similar to how viewpoints of structure-to-function have evolved, protein complexes are believed to be formed by different mechanisms. In the past, the lock-and-key mechanism (Fischer, 1894) was the widely accepted concept, and it dominated the design philosophies of in silico programs and in vivo protocols regarding protein docking and antibody drug design. However, as protein flexibility, conformational change and, most recently intrinsic disorder have received more attention, some researchers are shifting their foci on the induced-fit mechanism (Koshland, 1958). In contrast to lock-and-key, which assumes a more rigid structure of interacting proteins, induced-fit suggests that proteins adjust their shapes and forms when they approach each other to eventually form a complex.

Predictions of the protein-protein interface region can be categorized in terms of the features they use. The first category of predictions is based only on protein-sequence information. Methods in the second category use structural information to refine sequence sets that are then used to construct predictors. The last category of methods uses 3D structure information, or a combination of 3D structure and sequence for prediction. Various learning algorithms have been used for this purpose, including support vector machines (SVMs), neural networks, and others.

Since a significant number of complexes have large enough interfaces, we hypothesize that the complex formation follows the induced-fit mechanism rather than the lock-and-key mechanism. Therefore, one should be able to characterize interface regions by frequent appearances of unstructured or flexible amino acid residues in those regions. We believe the process whereby proteins form complexes is very similar to the late stages of protein folding; hence, some residues in the interface region undergo a transition from disorder to order and take their functional form.

In this article, we proposed a novel predictive model for identifying protein interaction sites, based on structural disorder and flexibility, as indicated in our hypothesis. Furthermore, we applied a tree-decomposition approach combined with support vector machines (SVMs) to overcome the running time problem that is commonly encountered in binding site prediction.

SVMs have proven very effective for solving pattern classification problems. However, it could be quite slow to fine tune the parameters in training. Hence, there is an ongoing effort to speed up SVM training. We adopted the so-called "data-reduction" approach to reduce a large training data set to many smaller data sets. Previously, researchers used bagging, boosting and divide-and-combine strategy for data-reduction.

“Tree decomposition support vector machine” (TDSVM) is the method we proposed to handle problems created by large data sets, such as the one we have in hand, to deal with protein-protein interfaces. TDSVM uses a decision tree to decompose the data space and then trains local SVMs on the decomposed regions. The decomposition approach reduces the total training time for the following reason. The time complexity of training an SVM is in the order of n^2 when the number of training samples is n. If each local SVM deals with σ samples, where $\sigma < n$, then the complexity of solving all the local problems is in the order of $(n/\sigma)\times\sigma^2 = n\sigma$, which is much smaller than n^2 if n is significantly larger than σ. The role of the decision tree as a decomposition scheme has some further advantages for large-scale SVM training. First, it can classify some data points by its own means, thereby reducing the cost of training SVMs on the remaining data points. Second, it is efficient in

finding the parameter values that maximize the performance score in the validation process which, helps achieve good results in the test process. Third, we can provide theoretical bounds for the generalization error of the classifier derived by the tree decomposition method.

For experimental data sets whose size can be handled by current kernel-based SVM training techniques, the proposed method can speed up the training by a factor of thousands, and still achieve comparable test accuracy. For much larger data sets such as the one we adopted in protein-protein interface prediction, it can accomplish training within one and half days on a computing environment equipped with Intel Xeon CPU 1.6GHz with a 6GB RAM**.** TDSVM works efficiently and effectively for our application because local SVMs are built on leaves whose size does not exceed 1,500. Another reason why TDSVM might work faster than other techniques is that, homogenous leaves (each such leaf in a decision tree has just one classification label) do not need further classification. While having many homogenous leaves is not the main reason for the efficiency of our prediction method, it was a key factor in previous applications of TDSVM.

Our contribution in this paper is twofold. First, we have proposed a tree decomposition approach to speedup SVM training. The resultant TDSVM classifier can be constructed in a much shorter time than the corresponding global SVM (gSVM) classifier that is trained on the whole data space. At a computing speed of three hundred times faster than gSVM, TDSVM enables us to conduct numerous experiments that would otherwise take months.

Second, our results show that we can predict protein-protein interface region with high performance scores using only sequence information. The results also tend to support our hypothesis on the induced-fit mechanism; the corresponding features that we extracted based on this hypothesis work quite well for prediction purposes. An even better result is expected with the addition of structural information to our machine learning process, if we continue this endeavor in the future.

In TDSVM we have a tool for the prediction of interaction sites that can be valuable as a first approach for guiding experimental methods investigating protein–protein interactions and localizing the specific interface residues. This tool will be valuable for scientists working on this problem either in silico or in vivo.

2. Data set

We derived our data set from the database "PROTCOM" (Kundrotas and Alexov, 2007). The portion we adopted comprised of known 3D structures of protein–protein complexes in the Protein Data Bank. The database contains 1,350 two-chain protein hetero-complexes, 7,773 homodimers, and some entries constructed from PDB files for multi-chain protein complexes by leaving only two interacting chains.

The proteins in "PROTCOM" are selected from X-ray structures, and satisfy the following criteria:

1. The sequence identity between any two entries must be <95%. This criterion is also applied to the sequences belonging to the same protein–protein complex.
2. The area of the complex interface must be >250 $Å^2$ and cover <50% of surface accessible area of either component.

3. There should be at least two secondary structural elements (helix or strand) in each component of the complex.
4. Very often, X-ray PDB files of two-chain complexes contain several identical pairs of proteins, which are artifacts of crystallization. If this is the case, the "PROTCOM" database only includes one pair.

We began by extracting all homodimers and heterodimers from "PROTCOM". Then, we removed from this collection those protein complexes with more than 40% sequence identity, following the standard practice of reducing the redundancy from our training samples. We also removed the complexes whose interface areas are less than 1000 $Å^2$. The reason for such a size constraint is that the induced-fit mechanism and intrinsic disorder are more prominent in larger interface areas. As a result, we obtained 4,064 protein chains.

The features were derived from two sources. The first source is AA-index (Kawashima et al., 2008), which is a compilation of physicochemical characteristic values for the twenty amino acids. We did not choose all the indices as there are more than 500 of them. Instead, we chose the "average flexibility index", which, we believe, manifests the induced-fit mechanism rather accurately. The second source is VLS2 (Obradovic et al., 2005; Peng et al., 2006), the predictor on protein disorder, proposed by A. K. Dunker's pioneering research group. VLS2 has the capability of predicting protein disorder by considering amino acid composition and other physicochemical properties.

We constructed our feature vectors in two steps. First, we calculated both features (from AA_index and VSL2) for each amino acid residue in our data set. Then, we applied a standard sliding window approach and arbitrarily choose window size 7. In other words, we considered a residue and three of its immediate sequential neighbors on both sides of the protein primary sequence. Therefore, for each residue, we constructed a feature vector of dimension 14 (2×7). As a result of processing 4,064 protein chains, we derived a data set comprising 1,239,814 fourteen-dimensional feature vectors.

3. Method

In this section, we briefly describe the TDSVM method. A complete description of the method can be found in Chang et al. (2009), which details the learning algorithm, the experiments, the comparison with alternative methods, and also a theoretical bound for the generalization error of the method. The implementation of TDSVM is available at

http://ocrlnx03.iis.sinica.edu.tw/~dar/Download%20area/tdsvm.php3,

where one can find the source code, the execution file, and a few exemplars.

TDSVM first trains a binary tree to decompose a given data space into small regions; then, it trains local SVMs on the decomposed regions. To describe the learning algorithm of TDSVM, we divide our discussion into two parts: 1) the training of a binary decision tree; and 2) the training of local SVMs and the search for optimal values of the parameters involved in the training.

3.1 Training a Binary Decision Tree

For the decomposition scheme, we adopt CART (Breiman et al., 1984) or binary C4.5 (Quinlan, 1986), which, allows two child nodes to grow from each node that is not a leaf.

We assume that all samples are represented as a d-dimensional feature vector whose class type is specified by a label y. To train a binary tree, we start with the root that takes all the training samples as input. We then decide whether to send each sample to the left-hand or right-hand child node. The same procedure is repeated for each node and its child nodes in a recursive manner.

At a given node E, we pick a feature f_E and a split point v_E so that all elements of E with $f_E < v_E$ are sent to the left-hand child node, and the remaining elements are sent to the right-hand child node. The values of f_E and v_E are determined as follows.

For each feature f, we define the split point v_f, associated with f, as the value v that maximizes the information gain $IG_E(f, v)$. We then choose feature f_E as the feature f that maximizes $IG_E(f, v_f)$ and v_E as the split point associated with f_E.

Intuitively, the information gain IG_E measures how much uncertainty is reduced by sending elements of E to the two child nodes of E. To quantify the uncertainty of a set S of labeled samples, we use the following measure

$$U(S) = -\sum_y p(S_y) \log p(S_y),$$

where $p(S_y)$ is the proportion of S's samples labeled y. The information gain is then defined as

$$IG_E(f, v) = U(E) - \frac{|E_L|}{|E|} U(E_L) - \frac{|E_R|}{|E|} U(E_R),$$

where E_L consists of the elements of E with $f < v$, E_R consists of the remaining elements of E, and $|S|$ is the size of S for any set S.

We stop splitting a node E when one of following conditions is satisfied: (i) the number of samples that flow to E is less than a *ceiling size* σ; or (ii) when $IG_E(f, v) = 0$ for all f and v at E. The value of σ in the first condition is determined in a data-driven fashion, which we describe in Section 3.2. The second condition occurs when all the samples that flow to E are homogeneous or when a subset of them is homogeneous and the remaining samples, although carrying different labels, are identical to some members of the homogeneous subset. There are other possible cases for the second condition, but as they rarely occur in practice, they do not concern us here.

3.2 Training local SVMs

After growing a tree, we train a local SVM on each of its leaves, using samples that flow to each leaf as training data. A tree and all local SVMs associated with its leaves constitute a TDSVM classifier (Figure 1).

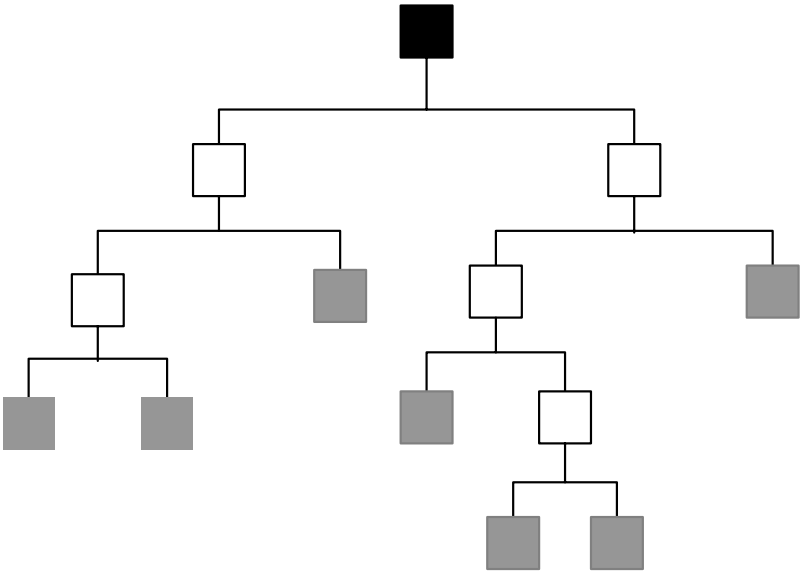

Figure 1. A TDSVM classifier comprises i) a tree with a root (the dark square), leaves (the gray squares), and the remaining nodes (the white squares), and ii) local SVMs associated with the leaves.

The parameters associated with a TDSVM classifier are: (i) σ, the ceiling size of the decision tree; and (ii) the SVM-parameters. All the local SVMs in a TDSVM classifier take the same SVM-parameter values. The optimal values of all the above parameters are determined as follows. Assuming that a training data set and a validation data set are given, we build TDSVM classifiers on the training data set and determine the optimal parameter values with the help of the validation data set. The training process proceeds as follows.

In the initial stage, we train a binary tree with an *initial* ceiling size σ_0, and then train all the local SVMs with the same SVM-parameters $\boldsymbol{\theta}$. Note that we express $\boldsymbol{\theta}$ in boldface to indicate that it may consist of more than one parameter. Let $v(\sigma_0, \boldsymbol{\theta})$ be the performance score of the resultant TDSVM classifier, measured on the validation data set. Define

$$\boldsymbol{\theta}_0 = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}}\, v(\sigma_0, \boldsymbol{\theta}) \text{ , and}$$

$$b(0) = v(\sigma_0, \boldsymbol{\theta}_0),$$

where Θ is the set of all possible SVM-parameter values whose effects we want to evaluate. The value $b(0)$ is the best performance score of all TDSVM classifiers with the ceiling size σ_0. In our experiments, we set $\sigma_0 = 1{,}500$.

In the subsequent stages, we construct TDSVM classifiers with a larger ceiling size, but we only train their local SVMs with top-ranked $\boldsymbol{\theta}$. To do this, we rank $\boldsymbol{\theta}$ in descending order of $v(\sigma_0, \boldsymbol{\theta})$. Let $\Theta_{[k]}$ be the set that consists of k top-ranked $\boldsymbol{\theta}$. In our experiments, we set k to 5.

More specifically, in stage t, we set $\sigma_t = 4\sigma_{t-1}$ for $t = 1, 2, \ldots$. We modify the tree with ceiling size σ_{t-1} by dropping a few nodes from the lower levels so that the ceiling size becomes σ_t. Then, we train new TDSVM classifiers on the modified tree with $\boldsymbol{\theta}$ chosen from $\Theta_{[k]}$. Define

$$\boldsymbol{\theta}_t = \underset{\boldsymbol{\theta}\in\Theta_{[k]}}{\operatorname{argmax}}\, v(\sigma_t, \boldsymbol{\theta}), \text{ and}$$

$$b(t) = v(\sigma_t, \boldsymbol{\theta}_t).$$

The value $b(t)$ is the best performance score of all TDSVM classifiers with the ceiling size σ_t. We terminate the process when the improvement in the best performance score is insignificant or we have reached the root node of the tree.

The steps of the TDSVM training process are as follows.

1. Set $t = 0$. We train TDSVM classifiers with the ceiling size σ_0 and the SVM-parameters $\boldsymbol{\theta}$ chosen from Θ. Then, we compute $b(0)$.
2. Increase t by 1 and set $\sigma_t = 4\sigma_{t-1}$. We obtain a binary tree with ceiling size σ_t and train TDSVM classifiers on that tree with the ceiling size σ_t and SVM-parameters $\boldsymbol{\theta}$ chosen from $\Theta_{[k]}$. Then, we compute $b(t)$.
3. If $b(t) - b(t\text{-}1) < thrld$, or σ_t exceeds the size of the training-data, we terminate the process; otherwise, we return to step 2. Note that when we terminate the process, we output the TDSVM classifier with the ceiling size σ_τ and SVM-parameters $\boldsymbol{\theta}_\tau$, where $\tau = t\text{-}1$ if $b(t) - b(t\text{-}1) < thrld$; or $\tau = t$, otherwise.

In step 3, the value of *thrld* must be set according to the type of performance score used. Usually, when the score is a real number between 0 and 1, the *thrld* is set to 0.005. We provide examples of this type of score in Section 4.

In summary, a TDSVM classifier is composed of a binary tree and a number of local SVMs, each of which is trained on a leaf of the tree. In the training process, we search for the best ceiling size and the optimal values of the SVM-parameters taken by all the local SVMs. We conduct the search in an iterative manner. In the first stage, we set an initial ceiling size and examine all possible values of the SVM-parameters. Then, in the subsequent stages, we quadruple the ceiling size in each stage and only examine the k top-ranked SVM-parameters that were determined in the initial stage.

4. Experiment Results

We conducted our experiments as follows. We randomly divided the data set into three components: training, validation, and test components, in a ratio of 4:1:1. We built TDSVM classifiers on the training component. Following the standard procedure, we normalized each feature vector to a vector of values between 0 and 1. To save time, we took *one-against-one* training mode (Knerr et al., 1990). The local SVMs in the TDSVM classifiers were kernel-based SVMs. Moreover, we used the RBF kernel function to measure the similarity between vectors. As a result, we had two SVM parameters: the penalty factor C, whose values were taken as $\Phi = \{10^a: a = \text{-}1, 0, \ldots, 5\}$; and the γ parameter in the RBF function, whose values were taken as $\Psi = \{10^b: b = \text{-}4, \text{-}3, \ldots, 4\}$. Thus, the set of all SVM parameter values was $\Theta = \Phi \times \Psi$. We then used the validation component to find the optimal parameter values. Finally, we applied the TDSVM classifier trained with the optimal parameter values to the test component to obtain a test accuracy rate.

Quantities used to define the performance scores for TDSVM are:

- TP - the number of true positives, which are residues correctly classified as interface residues;
- TN - the number of true negatives, which are residues correctly classified as non-interface residues;
- FP - the number of false positives, which are non-interface residues incorrectly classified as interface residues;
- FN - the number of false negatives, which are interface residues incorrectly classified as non-interface residues.

Based on the above definitions, we have:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}},\quad \text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FN}},\quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}},\quad \text{F}_1 = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}},$$

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FN}+\text{FP}+\text{FN}}, \text{ and}$$

$$\text{MCC} = \frac{\text{TP}\times\text{TN} - \text{FP}\times\text{FN}}{\sqrt{(\text{TP}+\text{FN})(\text{TP}+\text{FP})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}.$$

Our data set is imbalanced, with a ratio of 1:7 favoring negative examples (i.e., those amino acids that do not occur in interface regions). Therefore, when training the TDSVM, we optimized parameter values based on the MCC score, which is generally regarded as a balanced measure for classes of very different sizes. In the testing process, we derived all scores from the resultant classifier.

4.1 TDSVM versus gSVM

Previously, we tried gSVM for our classification task. However, using a personal computer, it took almost one week to train gSVM on one set of parameter values, while we needed to train it on 63 sets of parameter values. We then decided to utilize TDSVM, which accomplished the training of all the sets within one and half days. Thus, the estimated speedup factor of TDSVM was around 300.

TDSVM works efficiently and effectively for our application because local SVMs are built on leaves whose sizes do not exceed 1,500. In addition, TDSVM may work faster than other speedup techniques when there are many homogenous leaves. This situation does not occur in our application, since the proportion of training samples falling within homogenous leaves is very low (about 0.07%).

4.2 Experiment Results on Independent Test Data Sets

A number of works in the literature only reported their performances on their own independent test data sets, which makes a comparison with our results difficult. In Table 1, we list the results of some of those works, along with our results.

	Precision	Specificity	Recall	F_1-measure	Accuracy	MCC
TDSVM	85.8%	98.88%	47.03%	60.76%	92.36%	0.6009
Chung et al., 2006	50%	N/A	67.3%	57.37%	N/A	N/A
Chen et al., 2005	55%	N/A	51%	52.92%	N/A	N/A
Fariselli et al., 2002	72%	85%	56%	63.00%	73%	N/A
Wang et al., 2006	50.4%	N/A	65%	56.78%	N/A	N/A
Chen and Jeong, 2005	N/A	N/A	N/A	N/A	N/A	0.28
Li et al., 2007	53.6%	N/A	59.5%	56.4%	69.2%	0.328

Table 1. The performance scores derived from various works, each of which uses an independent test data set. (N/A means that the corresponding figures were not produced.)

4.3 Experiment Results on A Benchmark Data Set

We also tested our TDSVM classifier on a benchmark data set so that we could make a fair comparison with some previous results. This benchmark set is called Protein-Protein docking benchmark 2 provided by (Mintseries et al., 2005). There is now an updated version of Protein-protein docking benchmark (version 3). Moreover, since no other works (to the best of our knowledge) use it as their benchmark, we chose to use version 2 as well. Note that in applying TDSVM to the benchmark data set, we first removed the samples in Protein-Protein docking benchmark 2 from our training and validation data sets. We then trained a new TDSVM classifier on the modified data set. By so doing, we avoided including the same samples in both the learning and testing processes. In Table 2, we show all performance scores of all the works using the same benchmark data set.

	Precision	Specificity	Recall	F_1-measure	Accuracy	MCC
TDSVM	74.39%	97.17%	56.07%	63.94%	91.92%	0.6025
Liang et al., 2006	37.5%	N/A	36.3%	36.89%	N/A	N/A
Dong et al., 2007	38.3%	N/A	40.9%	39.56%	N/A	N/A
de Vries et al., 2006	34.3%	N/A	30%	32.01%	N/A	N/A

Table 2. Comparison of the performance scores derived from various works that use the same test data set, but different training and validation data sets. (N/A means that the corresponding figures were not produced.)

5. Discussion

We acknowledged the importance of understanding protein-protein recognition in our research from the outset. As stated previously in this paper, two mechanisms may be responsible for protein-protein recognition, namely, the lock-and-key mechanism and the induced-fit mechanism. Classical docking methods focus on smaller interfaces and the lock-and-key mechanism. Compared to induced-fit, lock-and-key implies a rigid-body type of protein-protein recognition, which yields a small or a standard-size interface. Lock-and-key was, and still is, the paradigm of “wet-lab” and “dry-lab” research on protein docking and drug design. However, one cannot ignore emerging re-

search on protein flexibility, intrinsic disorder and its impact on the understanding of protein complexes. These works lead us to the induced-fit mechanism, which suggests that proteins adjust their shape and form when they approach each other and eventually form a complex. Larger interfaces are formed by the "induced fit" mechanism, and disorder plays an important role in the mechanism. The following example (Figure 1) shows how enzymes function according to the induced fit mechanism hypothesis.

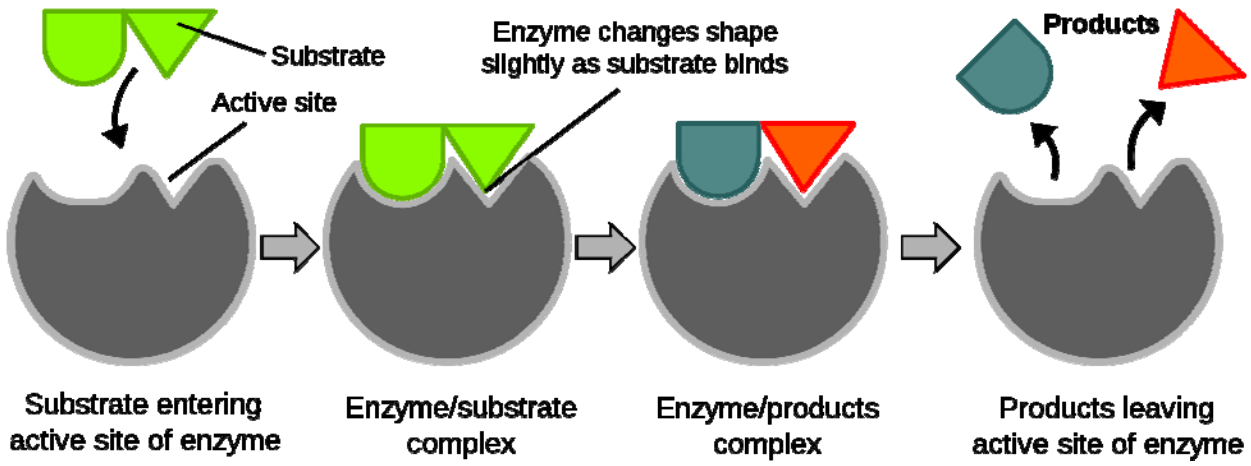

Figure 1. Diagrams to show the induced fit hypothesis of enzyme action, taken from http://en.wikipedia.org/wiki/Enzyme)

The induced-fit mechanism motivated our hypothesis. The results support the hypothesis and demonstrate that intrinsic disorder and protein flexibility are very useful information in protein-protein interface prediction.

Our results also show that we can predict protein-protein interface regions at high performance scores using only the sequence information. This would not be possible without the novel method that we have proposed, namely, TDSVM. It saves a tremendous amount of time by training local SVMs on decomposed regions. Moreover, it helps us produce better performance scores, since we can exploit a more comprehensive data set than those employed by other research teams.

6. Conclusion

We have proposed a tool for the prediction of interaction sites that can be used as a first approach for guiding experimental methods investigating protein–protein interactions and localizing the specific interface residues. We believe the tool will be valuable for scientists working on this problem either in silico or in vivo.

In the future we plan to expand the research reported in this paper. One possible direction would be to add more features, such as real protein structural information, to our approach, in order to improve our method's performance. Another direction would be to consider related topics, in particular, more specific protein binding problems, where we believe our research efforts would also be fruitful.

7. Supplementary Materials

The three data components, along with an execution file that can run TDSVM to produce the

reported results are available from the following website.

http://ocrlnx03.iis.sinica.edu.tw/~dar/Download%20area/dataset_ppi.php3.

References

Bahadur, R. P. and Zacharias, M. (2008) The interface of protein-protein complexes: Analysis of contacts and prediction of interactions. *Cell Mol Life Sci*., 65(7-8):1059-72.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) *Classification and Regression Trees*. Chapman and Hall.

Chang, F., Guo, C.-Y., Lin, X.-R., and Lu, C.- J. (2009) Tree decomposition for large-scale SVMs: experimental and theoretical results. Technical Report, Number TR-IIS-09-002, Institute of Information Science, Academia Sinica, 2009.

Chen, X. and Jeong, J. C. (2009) Sequence-based prediction of protein interaction sites with an integrative method. *Bioinformatics*, 25(5):585-591.

Chen, H. and Zhou, H. X. (2005) Prediction of interface residues in protein-protein complexes by a consensus neural network method: Test against NMR data. *Proteins*, 61:21-35.

Chung, J. L., Wang, W., and Bourne, P. E. (2006) Exploiting Sequence and Structure Homology to Identify Protein-Protein Binding Sites. *Proteins: Structure, Function and Bioinformatics*, 62(3) :630-640.

Kleanthous, C. (2000) *Protein-protein recognition*, Oxford University Press.

Dong, Q., Wang, X., Lin, L., and Guan, Y. (2007) Exploiting residue-level and profile-level interface propensities for usage in binding sites prediction of proteins. *BMC Bioinformatics*, 8:147.

Fariselli, P., Pazos, F., Valencia, A., and Casadio, R. (2002) Prediction of protein--protein interaction sites in heterocomplexes with neural networks. *Eur. J. Biochem*., 269:1356-61.

Fischer E. (1894). "Einfluss der Configuration auf die Wirkung der Enzyme". *Ber. Dt.* Chem. Ges., 27: 2985–93.

Fong, J. H., Shoemaker, B. A., Garbuzynskiy, S. O., Lobanov, M. Y., Galzitskaya, O. V., et al. (2009) Intrinsic Disorder in Protein Interactions: Insights From a Comprehensive Structural Analysis. *PLoS Comput Biol*, 5(3): e1000316.

Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T., and Kanehisa, M. (2008) AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res.*, 36: D202-D205.

Koshland, D. E. (1958). Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proc. Natl. Acad. Sci.*, 44(2): 98–104.

Knerr, S., Personnaz, L., and Dreyfus, G. (1990) Single-layer learning revisited: A stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*, Springer-Verlag.

Kundrotas, P. J. and Alexov, E. (2007) PROTCOM: searchable database of protein complexes enhanced with domain–domain structures. *Nucleic Acids Res.*, 35:D575-D579.

Li, M., Lin, L., Wang, X., and Liu, T. (2007) Protein–protein interaction site prediction based on conditional random fields. *Bioinformatics*, 23(5):597-604.

Liang, S., Zhang, C., Liu, S., and Zhou, Y. (2006) Protein binding site prediction using an empirical scoring function. *Nucleic Acids Res*., 34:3698-3707.

Mintseris, J., Wiehe, K., Pierce, B., Anderson, R., Chen, R., Janin, J., and Weng, Z. (2005) Protein-Protein Docking Benchmark 2.0: an update. *Proteins*, 60(2):214-6.

Obradovic, Z., Peng, K., Vucetic, S., Radivojac, P., and Dunker, A. K. (2005) Exploiting Heterogeneous Sequence Properties Improves Prediction of Protein Disorder, *Proteins*, 61(S7):176-182.

Peng, K., Radivojac, P., Vucetic, S., Dunker, A. K., and Obradovic, Z. (2006) Length-Dependent Prediction of Protein Intrinsic Disorder, *BMC Bioinformatics*, 7:208, 2006.

Quinlan, J. R. (1986) Induction of Decision Tree. *Machine Learning*, 1(1): 81-106.

de Vries, S. J., van Dijk, A. D., and Bonvin, A. M. (2006) WHISCY: What information does surface conservation yield? Application to data-driven docking. *Proteins*, 63:479-89.

Wang, B., Wong, H. S., and Huang, D. S. (2006) Inferring protein-protein interaction sites using residue conservation and evolutionary information. *Protein Pept. Lett*., 13:999-1005.