Discriminative Feature selection for Remote Homology detection using sparse coding

GROUP MEMBERS

SANJOY DEY
SAUPTIK DHAR
KAILASH RAMANATHAN

GUIDED BY

PROF. RUI KUANG
PROF. CHAD MYERS

Discriminative Feature selection for Remote Homology detection using sparse coding

ABSTRACT: Prediction of a protein's structural or functional class based on remote homology or the similarity of its amino acid sequence or structure to a protein whose function and structure is known is a fundamental problem in computational biology. There has been substantive amount of work towards using discriminative learning algorithms, in particular support vector machines (SVMs), for classification of proteins. The ulterior motives in most of these works were to suggest methodologies to provide a fixed-vector representation for the proteins or to incorporate priori knowledge using different biologically motivated Kernels. In this work we take a step further in using the fixed-vector representations (or Kernel representations) of the proteins already available and instead of applying them directly to the classifier further boost the discriminative features that will help in building a better classifier. Finally we compare our method with the base representation and try to provide some biological validation for the better performance of this approach.

Keywords: Remote homology, Support Vector Machines, Pairwise Sequence Comparison and SCOP database, Sparse Coding.

Supplementary Materials: http://www.ece.umn.edu/users/dharx007/FuncGen_Project.htm

1. INTRODUCTION:

1.1 Predicting the structure and hence the function of a protein is a fundamental requirement for understanding the manner in which they participate in the various biochemical processes of a cell. They are several techniques used to solve this problem some of which are listed in [1]. We use an approach based on remote homology to address this problem.

Proteins are said to be homologous when they evolve from the same ancestral protein. Therefore the problem of remote homology detection is to indentify proteins that have such a common lineage. One way to say two proteins are homologous is assess the similarity that they have in their sequence or structure and function since proteins branching from the same ancestral protein are expected to share some properties in common between them. Detecting homology is a non-trivial problem. Section 1.2 describes the remote homology. Section 1.3 discusses the problems related to remote homology detection.

1.2 The figure shown in this section illustrates the evolutionary relationship that exists between homologous proteins. Homologues take two forms namely Orthologues and Paralogues. Orthologues are homologous proteins that are formed owing to a speciation event.

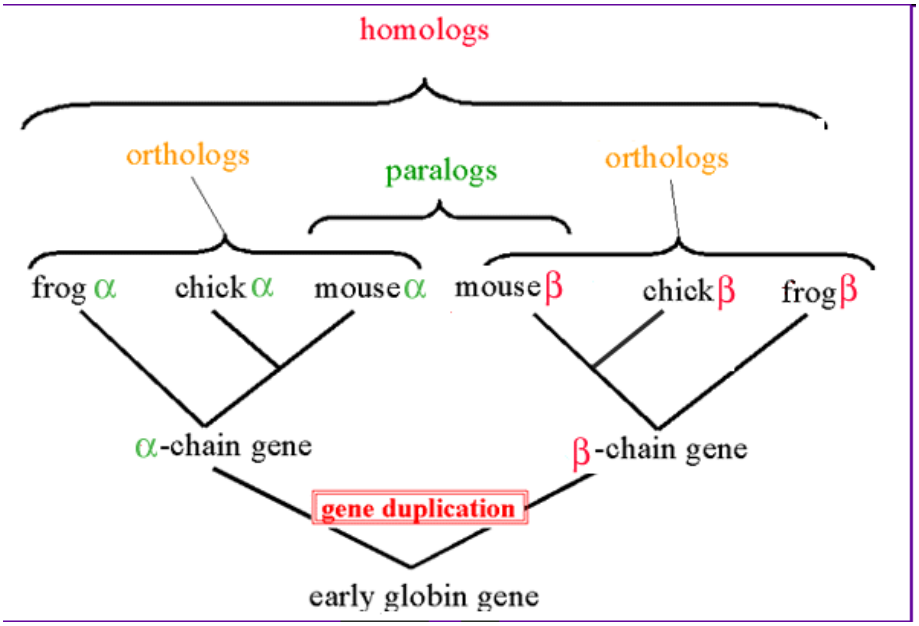

Figure 1.1: Relationship between Orthologues and Paralogues [28].

They therefore perform similar function in different species while paralogues are evolutionary related proteins that perform distinct functions. Paralogues are an outcome of a mutation event, insertions, deletions or substitutions, and they are generally found in the same genome. It must be noted that gene chick α is orthologous to mouse β and frog β but paralogous to chick β. Therefore, remote homology detection basically reduces to the problem of finding orthologues to given a newly sequenced protein whose function is unknown.

1.3 There are several challenges to detecting homology using similarity. First, we need a means to calculate the similarity between two sequences. The measure must faithfully reproduce the closeness relationship between genes/proteins in the evolutionary tree. By this we mean that the score should decrease as the height of the common ancestry increases from the level of comparison. Second, there can always be a similarity score calculated between two random input sequences to the algorithm. In other words does a high score warrant homology or does a small score imply that the sequences are non-related? This question is overcome by answering how statistically valid is the similarity and how much are they valid by chance. In order to overcome this problem we attach a significance value to the score calculated by any such measures. BLAST, a heuristic technique for calculating sequence similarity, overcomes this issue by appending an E-Value to the calculated score. The E-value says how much the related by chance; lesser the E-Value the better.
Issues such as convergent evolution and back evolution also causes issues when we calculate homology based on similarity. Section 2 talks about the data set used for the remote homology detection problem. Section 3 talks about the problem setup. In section 4 we provide the methods. Section 5 is the experimental setup and section 6 gives the results obtained using our method and the base model. In section 7 we provide an extension to our work by integrating the PPI dataset with the SCOP dataset. Finally we conclude in section 8 and provide our future directions in section 9.

2. SCOP database (date set)

We use the Structural Classification of Proteins (SCOP) dataset for our classification problem. The database follows a hierarchical categorization of proteins into classes, folds, superfamilies and families. These terms are described below:

- Family: Proteins that fall into the same class are functionally and structurally related proteins. They have clear sequence similarity and they are related by homology.
- Super-Family: These are protein sequences that have considerable structural and functional similarity but not that much sequence homology. They thus indicate divergent evolutionary relationship.
- Fold: These are sequences that have similar arrangement of secondary structures but not there is no evidence of evolutionary relationship.
- Class: These are proteins that have the same structural domain architecture.

The following table illustrates these relationships in the real data set. Only a part of the families are shown below:

id	Class	Fold	Superfamily	Family
1.27.1.1	All alpha proteins	4-helical cytokines	4-helical cytokines	Long-chain cytokines
1.27.1.2	All alpha proteins	4-helical cytokines	4-helical cytokines	Short-chain cytokines
1.36.1.2	All alpha proteins	lambda repressor-like DNA-binding domains	lambda repressor-like DNA-binding domains	Phage repressors
1.36.1.5	All alpha proteins	lambda repressor-like DNA-binding domains	lambda repressor-like DNA-binding domains	Bacterial repressors
1.4.1.1	All alpha proteins	DNA/RNA-binding 3-helical bundle	Homeodomain-like	Homeodomain
1.4.1.2	All alpha proteins	DNA/RNA-binding 3-helical bundle	Homeodomain-like	Recombinase DNA-binding domain
1.4.1.3	All alpha proteins	DNA/RNA-binding 3-helical bundle	Homeodomain-like	Myb

Each family (column 4) is designated by a family id. (column 1). We can see that different families fall under the same superfamily (column 5). Different superfamilies fall under the same fold(column 3) and different folds fall under the same class. This is the hierarchical organization of the SCOP database.

3. BASIC PROBLEM SETUP

For this project we use the common problem setup as used in [2],[3],[4]. For the sake of convenience we are reproducing the experimental setup. The main objective of our experiment is to asses the discriminative power of each method to classify the protein domains into super-families. So for each family the protein domains belonging to the family constitute the test samples and those outside the family but belonging to the same super-family are used as the training samples. Negative samples are taken from outside the positive sequence's fold and are randomly selected. A schematic representation of the setup is shown below:-

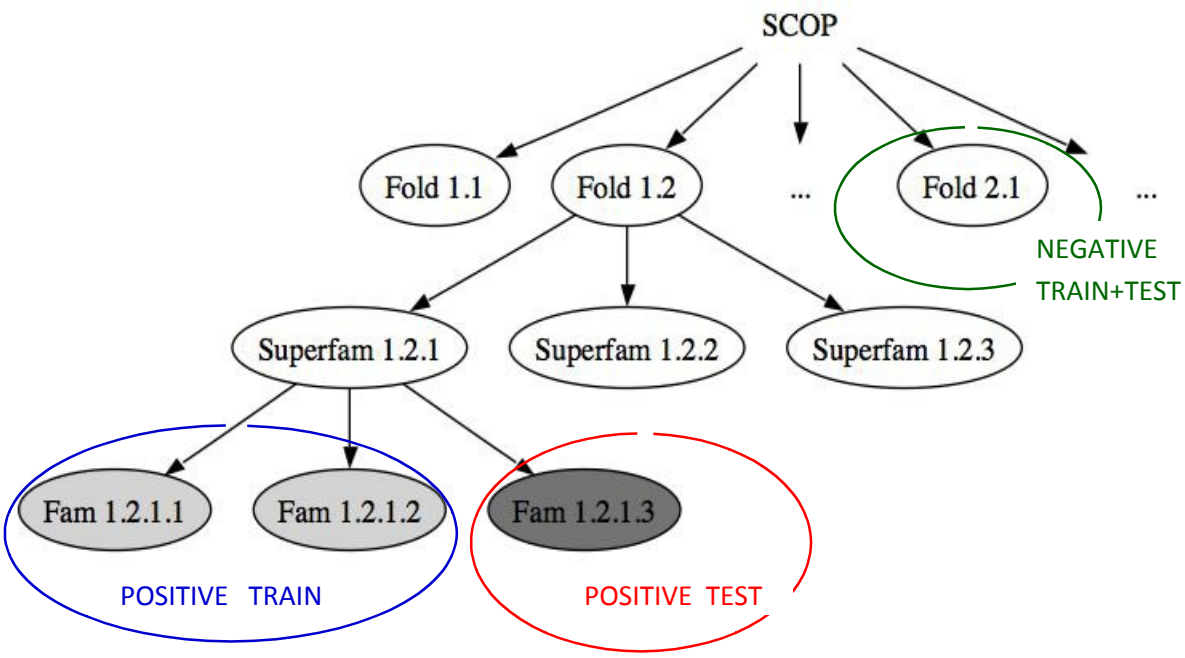

Fig 3.1 Basic problem setup for Family 1.2.1.3 [5]

4. METHODS

In this section we provide a brief explanation of our base model and our new model. This section is further sub-divided into 3 subsections. In section 3.1 we introduce a brief chronological development in remote homology detection and the current state of art. In section 3.2 we discuss about our baseline method and our newly proposed method and the motivation behind adopting the new model.

4.1 BRIEF CHRONOLOGICAL DEVELOPMENT

The development in the remote homology detection can be broken down to mainly 4 stages.

STAGE 1: In this stage the methodologies focused mainly towards representing the protein in a fixed vector form using the pair wise similarity between the proteins. Some of the examples include Smith-Waterman[6],BLAST[7] and FASTA[8]

STAGE 2: In this stage, further aggregate statistics from a set of similar sequences were used and the resulting statistics was used to a single unlabeled protein of interest. Profiles [9] and hidden Markov models (HMMs) [10] are two methods for representing these aggregate statistics.

STAGE 3: This stage marked the beginning of the introduction of semi-supervised learning, using the large databases of unlabeled protein sequences. Iterative methods such as PSI-BLAST[11]and SAM-T98[12] are some examples.

STAGE 4: A new era of homology detection was brought forward in this stage owing to Jaakkola[13]. He introduced the concept of using the negative samples in order to improve the discriminative power of SVM classifiers. This stage marks a rapid growth of research in Homology detection with the introduction of many marvelous methodologies. Some of them that highlight this list are: SVM pair wise, SVM Fischer, Mismatch string Kernel, Cluster Kernel etc. However the one method that really captures the most credit is the Profile-Kernel[14]. This is also the current state-of-art in this field. Recently, we have also seen a new method called Punting [4] .This method mainly tries to use two different classifiers,viz SVM and KNN classifiers and associates a confidence to the classifier classification. The classifier with the better confidence is more likely to predict correctly.

4.2 SVM PAIRWISE (BASE MODEL)

For this project we take the SVM PAIRWISE as the baseline model. The basic modeling of the SVM PAIRWISE Model can be given in the following steps:-

STEP 1: Construct a fixed length prototype representation of the protein sequences using the Smith-Waterman dynamic programming algorithm. (In this case both the positive and negative samples are taken for each of the families.)

STEP 2: Build an SVM model on this training set and use this classifier to predict on the test data.

The main strength of the SVM Pair wise lies in the usage of the negative samples in the fixed vector representation. This helps to increase the discriminative power of the SVM Classifier. A very good representation of the method can be given by the schematic diagram as provided in [2].

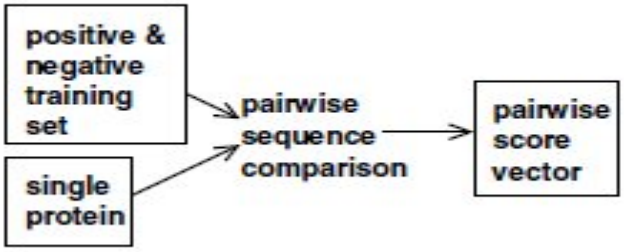

Fig 4.1: Schematic representation of the SVM Pair wise vector representation.

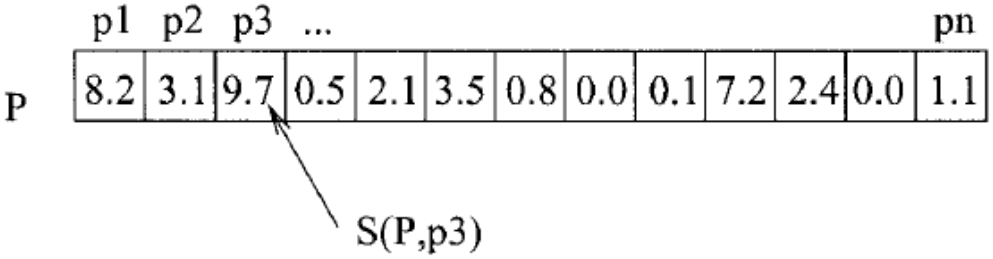

Fig 4.2: Representation of a possible pair wise score vector.

INTUITION (as provided in [2])

'The negative samples, if they contain significant similarity scores, can provide important evidence against a protein belonging to the positive class. For example, if a query protein is somewhat similar to sequences in the positive class but very similar to several proteins in the negative class, then the slight similarities to the positive class can safely be ignored. In the absence of these negative examples, the classification of such a sequence would remain in doubt.'

From Fig 4.1 and Fig 4.2 and the INTUITION provided above it makes sense to include the negative samples together with the positive samples to improve the discriminative power of the SVM algorithm. However, this leaves some new questions in the open.

- The families/super families are mainly organized based on the structural similarity. So does it make sense to include all the features?
- Is it not intuitive to include just a few features that are really discriminative between the different super-families?

An attempt to answer the question has led us to explore in the direction of feature selection. And our method marks the culmination of our search. (at least for now!)

4.3 SPARSE CODING (OUR APPROACH)

This section is further sub-divided into 2 sections. In section 4.3.1 we provide the biological motivation for sparse coding. In section 4.3.2 we provide the mathematical motivation and provide the mathematical setup for the experiments.

4.3.1 BIOLOGICAL MOTIVATION

'Mammalian brains consist of billions of neurons, each capable of independent electrical activity. Information in the brain is represented by the pattern of activation of this large neural population, forming a neural code. The neural code defines what pattern of neural activity corresponds to each represented information item. In the sensory system, such items may indicate the presence of a stimulus object or the value of some stimulus parameter, assuming that each time this item is represented the neural activity pattern will be the same or at least similar. One important and relatively simple property of this code is the fraction of neurons that are strongly active at any one time. Sparse coding is the representation of items by the strong activation of a relatively small set of neurons.' [15]

4.3.2 MATHEMATICAL SETUP

From the mathematical perspective sparse coding is the representation of a dense system by its sparse counterpart. To provide a better explanation on the system's perspective, let us consider that we have a system which has M equations and N variables. For a system with M=N the system can have only one unique solution. (Provided the system is non singular).On the other hand in case we have M>N then the system can have multiple solutions and is over- determined. So it makes sense to assume that for an over-determined system rather than a dense solution we can also expect a sparse equivalent. Our aim is to basically find the sparse equivalent by adding meaningful constraints. (Note by adding constraints we are effectively reducing the number of equations that are not active.)

Such a setup can be given in the form of the classical sparse coding formulation as:-

$$w* = \mathrm{argmin}_w \sum_i l\left(y_i, f(x_i, w)\right) \quad s.t \quad \|w\|_0^0 \leq L \qquad \text{(Equation 3.1)}$$

Here,
$l(.,.)$=The loss function.
w= parameter for the parametric function.
L= Threshold.
$\|w\|_0^0$= Pseudo-norm. (Indicates the sum of non-zero elements in the parameter)

Basically it is not possible to mathematically implement the Pseudo-norm. So there are a number of ways to relax the system. One such popular method is to use the L-1 norm. This converts the problem to a convex optimization problem. (Of course with a suitable loss function). There is also an alternative to use the L-2 norm and then to threshold with an optimal threshold. This would still induce sparsity; but there are a number of caveats in this approach. We explain them by using a simple experiment provided in [16].For the sake of convenience we reproduce the experiment setup and the figures involved.

In the experiment they take a matrix $A \in R^{100\times 30}$ and vector $b \in R^{100}$ (chosen at random, but the results are typical), and compute the $\ell 1$-norm and $\ell 2$-norm approximate solutions of $Ax \approx b$
From the plots of the penalty functions we note that:-

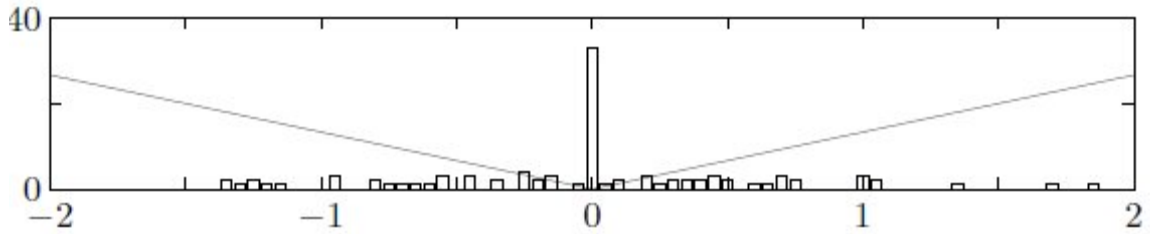

Fig 3.3 Histogram of residual amplitudes for L1 penalty function

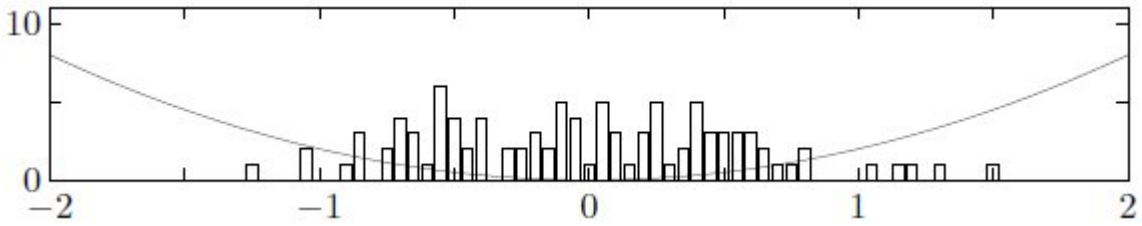

Fig 3.4 Histogram of residual amplitudes for L-2 penalty function

L1 penalization: For L1 penalization the samples (residual value) are mostly clustered near the origin with very sparse samples scattered away from origin.

L2 penalization: For L2 penalization the samples are mostly clustered on the range of -1 to 1.But the samples are mostly dense in this region.

Quiet conspicuous from the experiment is that the L1 norm is a better choice for inducing sparsity and is much easier to apply a threshold in that case as well.

Thus we relax the equation 3.1 to a more easily achievable equation provided below:-

$$w* = \arg min_w \sum_i l(y_i, f(x_i, w)) + \lambda \|w\|_1 \quad \text{(Equation 3.2)}$$

Here,

λ= The regularization parameter which strikes a balance between the degree of sparsity and the accuracy of the model.

Note: Here, rather than assuming a sparse equivalent of the dense w ;we impose sparsity by sacrificing the accuracy of the model.

Thus we use an L1 regularization factor to obtain a sparse parameter. Using this we can obtain a sparse prototype vector representation by using the algorithm below:-

ALGORITHM

STEP 1: Use a Fixed vector representation of the proteins.(Let the representation be given by X). We assume a model as

$$y \cong w^T * x \quad \text{(A least square solution)}$$

STEP 2: Perform supervised feature selection on the representations.(with sparse coding). Let,

$$f(x, w) = w^T * x$$

Thus, we have

$$w* = \arg min_w \|y - w^T * x\|_2 + \lambda \|w\|_1$$

STEP 3: Compute the relevant prototype representation.

$$a = x(i) \qquad s.t\ i \epsilon R$$

where,

$$R = \{i | w* > threshold\}$$

Thus we obtain the relevant prototype representation for the data and use that in building the SVM classifier.

5 EXPERIMENTAL SETUP

In this section we describe the experimental setup. Basically we follow the same experimental setup used by [2].For the sake of convenience we reproduce the experimental setup. For the project we use the SCOP 1.53 database [18].Sequences were selected using the Astral database (*astral.stanford.edu* [Brenner *et al.*, 2000]), removing similar sequences using an E-value threshold of 10^{-25}. The total number of samples was 4352. For the vectorization of the data we use the dataset readily available at [17]. This dataset has been constructed for SVM Pairwise and the basic vectorization step is provided in[2].However for the sake of convenience we reproduce the method used for vectorization of the protein sequences.

The vectorization step of SVM-pairwise uses the Smith–Waterman algorithm as implemented on the BioXLP hardware accelerator (*www.cgen.com*). The feature vector corresponding to protein P is, $P=p_1p_2p_3.....p_n$ where n is the total number of proteins in the training set and p_i is the E-value of the Smith–Waterman score between sequence P and the ith training set sequence. The default parameters used are: gap opening penalty and extension penalties of 11 and 1, respectively, and the BLOSUM 62 matrix.

For our project we use the readily available data (SW scores) provided in the website and further scale down the problem size for the ease of implementation and time constraint. Basically, we construct the dataset as:-

For each family
POSITIVE TRAINSET:- Protein domains belonging to the different families but to the same super-family.
POSITIVE TESTSET:- Protein domains belonging to the same families.
NEGATIVE TRAINSET & TESTSET: - Protein domains taken from outside the positive sequence's fold. These are randomly selected to be twice the number of the equivalent positive samples.

The entire dataset is available in [19].

In the following table 5.1 we provide the breakup of the number of samples for each Families.

BASE MODEL(SVM PAIR WISE)
For each of the families we build a linear SVM classifier on the training data and record the test error on the test data. We select the range of C values to be [0.001,0.01,0,1,1,10] and use the grid search for model selection. We use 5 Fold Cross Validation for the model selection.

SPARSE CODING
For this case we apply the training data to the ALGORITHM as given in Fig 3.5. For the algorithm we use a GRID search for the values of λ and threshold. The values of λ=[0.001 0.01 0.1] and the values of thereshold = [1e-5 1e-6 1e-7 1e-8 1e-9]. For all the 24 models we obtain the equivalent active index set (R).Then we select the x values of the trainset/testset corresponding to the active indices to get the prototype representation of the training data.

(Note: We do not use the x/y –values of the test set for getting the sparse representation.)
After this we use the sparse representation of the dataset and apply them to a linear SVM classifier with the similar parameters as used in the base model. The sparse coding is done using the YALMIP optimization toolbox interface with the SDPT3 solver [20],[21]. For the SVM Classifier we used the SMO interface available in the STPRTool[22]. All the codes are available in [19].

Family name	Positive train	Positive test	Negative train	Negative test	Family name	Positive train	Positive test	Negative train	Negative test
'7.3.5.2'	12	9	24	18	**'7.3.10.1'**	11	95	22	190
'2.56.1.2'	11	8	22	16	**'3.32.1.11'**	46	5	92	10
'3.1.8.1'	19	8	38	16	**'3.32.1.13'**	43	8	86	16
'3.1.8.3'	17	10	34	20	**'7.3.6.1'**	33	9	66	18
'1.27.1.1'	12	6	24	12	**'7.3.6.2'**	16	26	32	52
'1.27.1.2'	10	8	20	16	**'7.3.6.4'**	37	5	74	10
'3.42.1.1'	29	10	58	20	**'2.38.4.1'**	30	5	60	10
'1.45.1.2'	33	6	66	12	**'2.1.1.1'**	90	31	180	62
'1.4.1.1'	26	23	52	46	**'2.1.1.2'**	99	22	198	44
'2.9.1.2'	17	14	34	28	**'3.32.1.1'**	42	9	84	18
'1.4.1.2'	41	8	82	16	**'2.38.4.3'**	24	11	48	22
'2.9.1.3'	26	5	52	10	**'2.1.1.3'**	40	9	80	18
'1.4.1.3'	40	9	80	18	**'2.1.1.4'**	88	33	176	33
'2.44.1.2'	11	140	22	280	**'2.38.4.5'**	26	9	52	18
'2.9.1.4'	21	10	42	20	**'2.1.1.5'**	94	27	188	54
'3.42.1.5'	26	13	52	26	**'7.39.1.2'**	20	7	40	14
'3.2.1.2'	37	16	74	32	**'2.52.1.2'**	12	5	24	10
'3.42.1.8'	34	5	68	10	**'7.39.1.3'**	13	14	26	28
'3.2.1.3'	44	9	88	18	**'1.36.1.2'**	29	7	58	14
'3.2.1.4'	46	7	92	14	**'3.32.1.8'**	40	11	80	22
'3.2.1.5'	46	7	92	14	**'1.36.1.5'**	10	26	20	52
'3.2.1.6'	48	5	96	10	**'7.41.5.1'**	10	9	20	18
'2.28.1.1'	18	44	36	88	**'7.41.5.2'**	10	9	20	18
'3.3.1.2'	22	7	44	14	**'1.41.1.2'**	36	6	72	12
'3.2.1.7'	48	5	96	10	**'2.5.1.1'**	13	11	26	22
'2.28.1.3'	56	6	112	12	**'2.5.1.3'**	14	10	28	20
'3.3.1.5'	13	16	26	32	**'1.41.1.5'**	17	25	34	50

Table 5.1 Table of data break up for the different families.

6. RESULTS AND ANALYSIS

In this section we are going to describe different results that we got from the above methods. We applied our method with sparse coding for 54 families. Then we also applied the SVM-pairwise method on those 54 families to compare the classification error rates with that of our method. The comparative results are shown below in table 6.1 which shows that our method outperforms SVM-pairwise for most of the families (38 out of 54 families.)

Family Name	SVM pairwise	Sparse Coding	Family Name	SVM pairwise	Sparse Coding
'7.3.5.2'	0.11111	0.074074	'7.3.10.1'	0.094737	0.091228
'2.56.1.2'	0.33333	0.20833	'3.32.1.11'	0.13333	0.066667
'3.1.8.1'	0.041667	0.041667	'3.32.1.13'	0.25	0.125
'3.1.8.3'	0.13333	0.1	'7.3.6.1'	0.037037	0.037037
'1.27.1.1'	0.16667	0.16667	'7.3.6.2'	0.038462	0.012821
'1.27.1.2'	0.25	0.25	'7.3.6.4'	0.13333	0.066667
'3.42.1.1'	0.26667	0.16667	'2.38.4.1'	0.26667	0.13333
'1.45.1.2'	0.33333	0.33333	'2.1.1.1'	0.086022	0.064516
'1.4.1.1'	0.14493	0.14493	'2.1.1.2'	0.045455	0.030303
'2.9.1.2'	0.16667	0.14286	'3.32.1.1'	0.074074	0.037037
'1.4.1.2'	0.33333	0.20833	'2.38.4.3'	0.33333	0.27273
'2.9.1.3'	0.066667	0	'2.1.1.3'	0.14815	0.14815
'1.4.1.3'	0.14815	0.14815	'2.1.1.4'	0.18182	0.075758
'2.44.1.2'	0.35238	0.33333	'2.38.4.5'	0.074074	0.074074
'2.9.1.4'	0.066667	0.066667	'2.1.1.5'	0.32099	0.2963
'3.42.1.5'	0.33333	0.28205	'7.39.1.2'	0.095238	0
'3.2.1.2'	0.20833	0.1875	'2.52.1.2'	0.33333	0.26667
'3.42.1.8'	0.33333	0.26667	'7.39.1.3'	0.21429	0.11905
'3.2.1.3'	0.25926	0.22222	'1.36.1.2'	0.33333	0.33333
'3.2.1.4'	0	0	'3.32.1.8'	0.18182	0.18182
'3.2.1.5'	0.095238	0.095238	'1.36.1.5'	0.26923	0.064103
'3.2.1.6'	0.33333	0	'7.41.5.1'	0.33333	0.33333
'2.28.1.1'	0.33333	0.12879	'7.41.5.2'	0.33333	0.037037
'3.3.1.2'	0.14286	0.14286	'1.41.1.2'	0.11111	0.055556
'3.2.1.7'	0.4	0.26667	'2.5.1.1'	0.15152	0.060606
'2.28.1.3'	0.33333	0.27778	'2.5.1.3'	0.23333	0.13333
'3.3.1.5'	0.16667	0.041667	'1.41.1.5'	0.053333	0

Table 6.1 Table showing the Prediction error for the two different methods for the 54 Families.

Moreover, if we plot the curve for the error rate of SVM-pairwise Vs. SVM-sparse, we will get the following figure 6.1. In this figure, we can observe significant improvement in terms of classification accuracy (1-classification error) for 38 classes among 54 families and for rest of the 16 families the classification accuracy of our SVM-sparse was as good as the original SVM-pairwise.

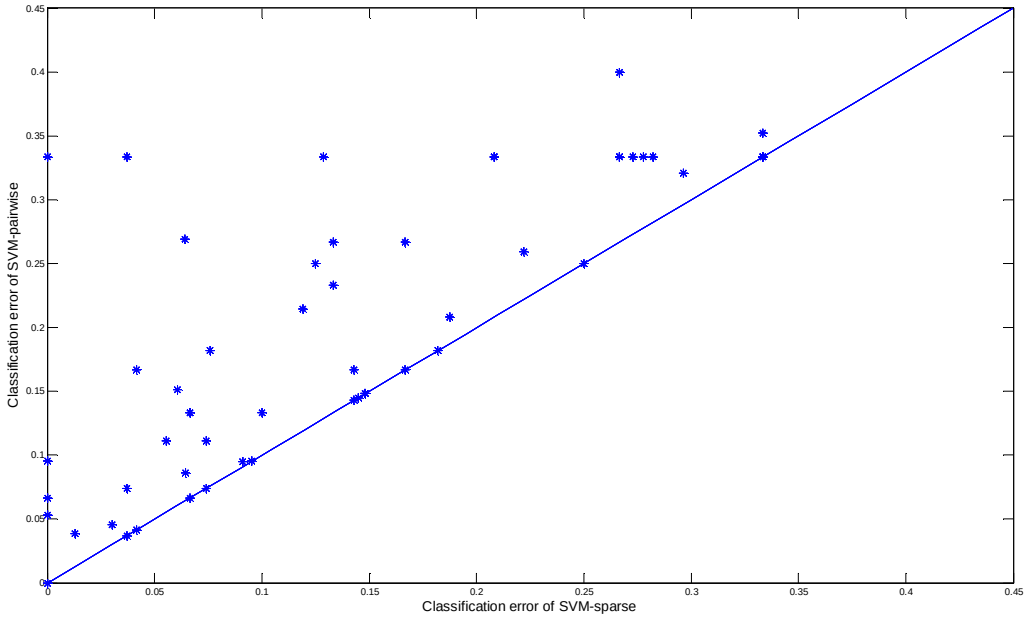

Figure 6.1: Classification error of SVM-pairwise vs. SVM-sparse

Moreover, we plotted the receiver operating characteristic (ROC) curve for our classifier. Here, we have chosen families 1.41.1.5, 2.1.1.1, 3.42.1.8 randomly for our purpose of analysis.

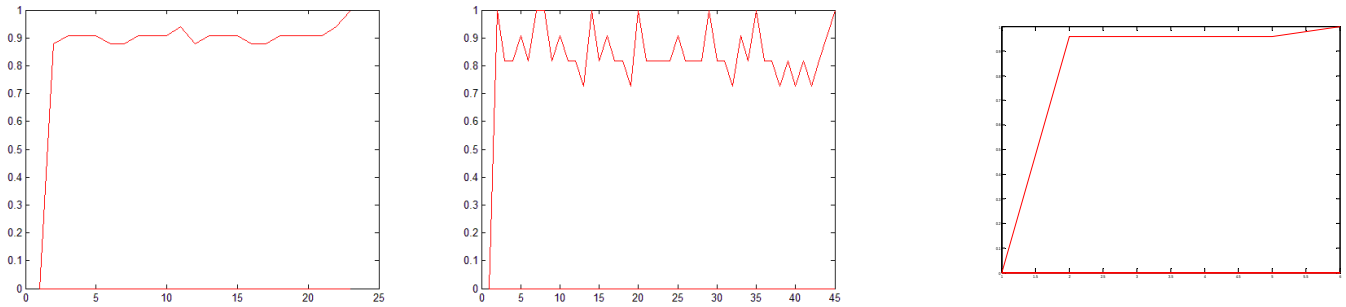

Figure 6.2: ROC curve for the family 2.1.1.1, 3.42.1.8 and 1.41.1.5 respectively

For all of these figures, the area under the ROC curve is close to one which shows that our ROC curves are non-random.

Moreover, in an attempt to understand the trend of the Prediction Error for the different threshold and lambda we plot the graph for the Family 3.42.1.5.

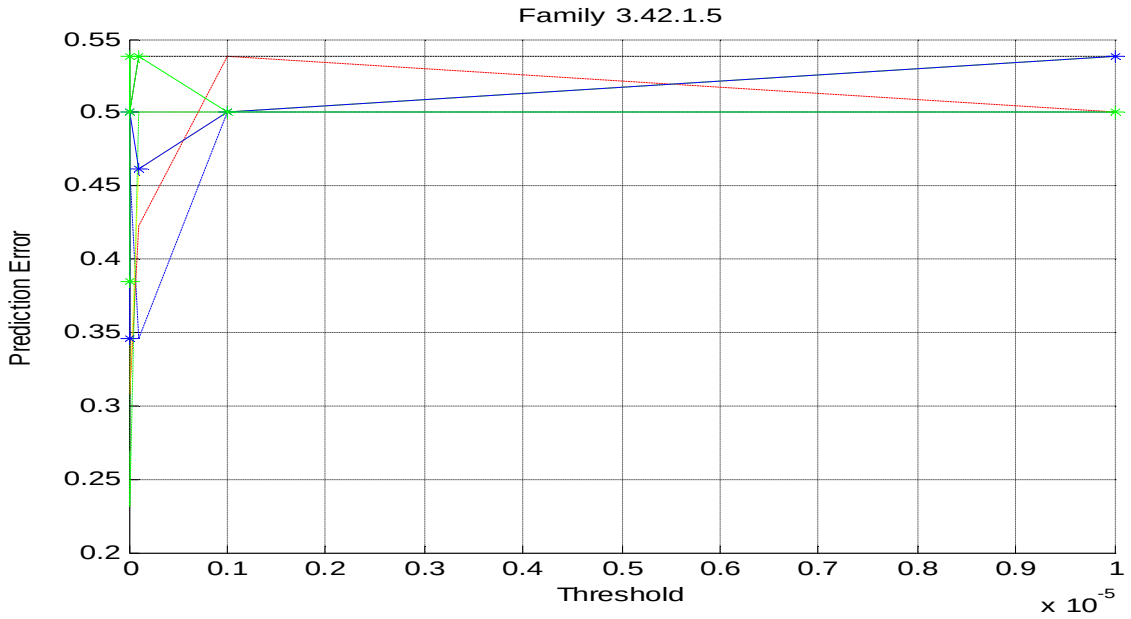

Fig 6.3 The plot of prediction error for different threshold

As we can see that with the increase with the threshold the Prediction error increases. Infact in this case we observe a trend that there is a certain dip in the prediction risk for a pretty small value of threshold(for all the λ values). Now smaller the threshold lesser is the number of features. This trend indicates that as we increase the number of features we lose the discriminative capacity. This further consolidates our approach to do feature selection with sparse coding.(Note: We have analysed this trend for just 4 Families. In order to make a claim we shall still need to generate the curve for all the families.)

7. Integrating Protein-Protein Interaction data with SCOP database:

To extend our current works further we tried to explore whether we can improve the classification accuracy further by integrating protein-protein integration (PPI) data with the SCOP database. The rationale behind merging the PPI dataset with SCOP dataset was that protein-protein interaction is highly related to the functions of the interacting protein [24]. Moreover, protein with similar structure are also supposed to have similar function to some extant. Hence, integrating PPI dataset might boost up the corresponding weight of the protein which has high sequence similarity and also interaction among them during the L1 minimization. However, the main problem we faced during integrating the PPI data with the SCOP data was that PPI data are species specific while SCOP dataset has protein sequences along multiple species. So, we kept our experiment mainly focused on human PPI network.

We found that there are 1113 of human proteins in SCOP 1.53 dataset and 553 proteins were selected for Smith-Waterman similarity dataset with E-values more than 10^{-25} threshold. Among these 553 proteins 542 proteins have the record for human protein-protein interactions in the HPRD database [25]. Moreover, as the names of the proteins were different in these two dataset, we have to convert first SCOP labels into SWISSPROT labels through the PDB dataset [26].

Here, we applied the disjunctive integration technique to augment the SCOP and PPI datasets and ended up with a dataset with 553 proteins and 4894 variables (4352 SCOP fixed vectors and 542 PPI interactions). Then we followed the whole processed described above on this new dataset. But as the human PPI network is very sparse, very few families ended up with significant amount of test and training data. Now if we apply SVM pair-wise and SVM-sparse coding on these new dataset, we found a little bit improvement for some families where we have some interactions in the newly augmented dataset. The accuracy is a little bit higher than the accuracy of the classifier that we built on single SCOP database.

The classification errors of the family 2.1.1.4 are given below as this family has largest training and test datasets with 367 and 146 samples respectively.

Method	Classification Error	Lamblda	Threshold	Active index
SVM-pairwise(on SCOP only)	0.1818	-	-	4352
SVM-sparse(on SCOP only)	0.076	0.01	1.00E-07	214
SVM-pairwise(on SCOP+PPI)	0.8767	-	-	4894
SVM-sparse(on SCOP+PPI)	0.068	1	1.00E-05	231

Table 7.1 : Classification errors of Family 2.1.1.4

Moreover, we tried to investigate the active indices of the 231 protein for which we got the best result. It turns out that among those 231 features, 2 of the proteins named ‘d1lcka1’ and 'd1lkka_' were selected from the PPI dataset and these two proteins have strong interactions in the dataset (Both proteins have 29 interactions where highest degree of the protein was 30). So, it seems reasonable that this proteins boost up the performances of our classifier. We applied such kind of for 3 more families and observed similar type of results for 2 families. The other family did not gain any performances from PPI data. We could not perform similar type of study for all families because, most of the dataset have very few samples for test and training data (<4).

In our presentation, we showed some wrong results where such kind of disjunctive integration showed lower accuracy than the single SCOP dataset. Basically there was an error in our preprocessing part and for that reason we got wrong results.

8. Discussion: Though it is a bit early to conclude. However we observe that the new method provide better results than the SVM Pairwise.(atleast for this particular case).This is just the initial step which was mainly directed towards a feasibility study of the concept. A further claim can be made only after using this method over the more better performing methods like the Profile Kernel, Cluster Kernel. Integrating PPI data with SCOP data gives some more accuracy improvements which necessitate further investigation.

9. ISSUES and FUTURE WORK

In this section we provide the issues pertaining to our model and the future direction of work that we intend to continue on this. We provide the point wise description for each.

ISSUES

1. The assumption of a linear model for the step 2 of the algorithm may result to some limitations.
2. The usage of L2 norm in the STEP2(as the loss function) may not be a good choice for classification. Empirical results show that minimization of the L2 norm does not necessarily mean minimization of the classification Error[23].This motivates the usage of Hinge loss in place of the L2 norm.
3. Some heuristic for selecting the λ and threshold parameter is also necessary for practical feasibility.

FUTURE WORKS

1. Providing a solid biological validation for the active indices is very necessary. Although we have observed the proteins (equivalent to the active indices) to belong to the same fold, we have not yet been able to provide a solid biological validation to consolidate out method. As such this tops the list of our priority for future works.
2. Integration of the PPI data for Function prediction based on Gene Ontology. Our empirical results suggest that merging the PPI data results into a better classification. A proper explanation as to why we observe such is also a necessity.
3. Moreover, disjunctive integration of PPI data may also imports noise into the SCOP dataset. So, other types of integration techniques can be explored. There are some methods like common neighborhood, H-confidence measures which remove noise from PPI data and make it to continuous data. So, those method can also be explored.
4. Provide empirical results for the parameter selection. It is computationally not feasible to do a grid search for the λ value. We are still looking for a clever heuristic. However we have not yet found any in current literature.

5. Usage of an appropriate Multi-Class classifier could also be a direction of work. There are multiple directions in which we can implement the Multi class classifier. Although it seems more enticing to use the multi-class SVM provided in http://svmlight.joachims.org/svm_multiclass.html . However not much can be said without some empirical results.

6. One more possible yet ambitious direction can be to use transfer learning for different classifiers. However identifying different groups still remains an intriguing problem.

REFERENCE

1. Gaurav Pandey, Vipin Kumar and Michael Steinbach, "Computational Approaches for Protein Function Prediction: A Survey", TR 06-028, Department of Computer Science and Engineering, University of Minnesota, Twin Cities.
2. LI LIAO and WILLIAM STAFFORD NOBLE, “Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships”, JOURNAL OF COMPUTATIONAL BIOLOGY,2003
3. J. Weston, C. Leslie, D. Zhou, A. Elisseeff and W. Stafford Noble. “Semi-Supervised Protein Classification using Cluster Kernels”. NIPS, 2003

4. I. Melvin, J. Weston, C. Leslie, W. S. Noble. “Combining classifiers for improved classification of proteins from sequence or structure”. BMC Bioinformatics.
5. I. Melvin, Ie, Eugene and Weston, Jason and Noble, William Stafford and Leslie, Christina: ‘Multi-class protein classification using adaptive codes’. Journal of Machine Learning Research, 2007.
6. Smith, T., and Waterman, M. 1981. Identi. cation of common molecular subsequences. *J. Mol. Biol.* 147, 195–197.
7. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search prorams. *Nucl. Acids Res.* 25, 3389–3402.
8. Pearson, W.R. 1985. Rapid and sensitive sequence comparisons with FASTP and FASTA. *Methods Enzymol.* 183,63–98.

9 . Gribskov, M., Lüthy, R., and Eisenberg, D. 1990. Pro. le analysis. *Methods Enzymol.* 183, 146–159.

10 . Krogh, A., Brown, M., Mian, I., Sjolander, K., and Haussler, D. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.* 235, 1501–1531.

11. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search prorams. *Nucl. Acids Res.* 25, 3389–3402.

12 . Karplus, K., Barrett, C., and Hughey, R. 1998. Hidden Markov models for detecting remote protein homologies.*Bioinformatics* 14(10), 846–856.

13 . Jaakkola, T., Diekhans, M., and Haussler, D. 2000. A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.* 7(1–2), 95–114.

14. Kuang, R., E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie (2005). Profile kernels for detecting remote protein homologs and discriminative motif. *Journal of Bioinformatics and Computational Biology*.

15. http://www.scholarpedia.org/article/Sparse_coding

16 . Convex Optimization by Stephen Boyd , Lieven Vandenberghe.

17 . http://www.ccls.columbia.edu/compbio/svm-pairwise/

18 . Brenner, S.E., Koehl, P., and Levitt, M. 2000. The ASTRAL compendium for sequence and structure analysis. *Nucl. Acids Res.* 28, 254–256.

19. http://www.ece.umn.edu/users/dharx007/FuncGen_Project.htm

20. http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php

21 . http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

22. http://cmp.felk.cvut.cz/cmp/software/stprtool/

23. Vladimir S. Cherkassky, Filip Mulier, ‘Learning from data: concepts, theory, and methods’, Wiley-IEEE, 2007

24. James C. Whisstock and Arthur M. Lesk , ‘Prediction of protein function from protein sequence and structure.’Cambridge Journal 2003.

25. www.hprd.org

26. http://www.rcsb.org/pdb/home/home.do