Subcellular localization for Gram Positive and Gram Negative Bacterial Proteins using Linear Interpolation Smoothing Model

Harsh Saini
saini_h@usp.ac.fj

Gaurav Raicar
raicar_g@usp.ac.fj

Abdollah Dehzangi
abdollah.dehzangi@griffithuni.edu.au

Sunil Lal
lal_s@usp.ac.fj

Alok Sharma
alok.fj@gmail.com

March 23, 2015

Abstract

Protein subcellular localization is an important topic in proteomics since it is related to a proteins overall function, help in the understanding of metabolic pathways, and in drug design and discovery. In this paper, a basic approximation technique from natural language processing called the linear interpolation smoothing model is applied for predicting protein subcellular localizations. The proposed approach extracts features from syntactical information in protein sequences to build probabilistic profiles using dependency models, which are used in linear interpolation to determine how likely is a sequence to belong to a particular subcellular location. This technique builds a statistical model based on maximum likelihood. It is able to deal effectively with high dimensionality that hinder other traditional classifiers such as Support Vector Machines or k-Nearest Neighbours without sacrificing performance. This approach has been evaluated by predicting subcellular localizations of Gram positive and Gram negative bacterial proteins.

1 Introduction

Subcellular localizations of proteins is very important research topic in molecular cell biology and proteomics since it is closely related to the protein's functions, metabolic pathways, signal transduction and other biological processes within a cell [1, 2]. Knowledge of a protein's subcellular localization also plays an important role in drug discovery, drug design and biomedical research. Determination of subcellular localizations experimentally is laborious, time-consuming and, in some cases, experimental means to determine some subcellular localizations of proteins is difficult using fluorescent microscopy imaging techniques [3].

In recent years, there has been significant progress in subcellular localization predication using computational means. There are approaches that extract features directly from the syntactical information present in protein sequences such as amino acid composition (AAC) [4, 5], N-terminus sequences [6] and pseudo-amino acid composition (PseAAC) [7]. Some approaches use the evolutionary information present in Position Specific Scoring Matrices (PSSM) to extract features [8]. Features can also be generated from protein databases such as the annotations in Gene Ontology (GO), functional domain information, and/or textual information from the keywords in Swiss-Prot [9–13]. Moreover, some researchers have utilized the information present in the physicochemical properties of amino acid residues to enhance prediction accuracy [4, 14]. However, most prevalent techniques are a hybrid collection of various features to help identify discriminatory information for the classifiers to obtain an improved prediction accuracy [4, 7, 9–12, 15].

In proteomics, frequencies or probabilities of occurrence for amino acid subsequences in proteins have been used to extensively model proteins. Some features that can be considered as variants of such models include Amino Acid Composition (AAC) [16], Pairwise Frequency (PF1) and Alternate Pairwise Frequency (PF2) [17], bigram [18], k-separated bigrams [19], and trigram [20]. Although such models have been rigorously studied by researchers, they have mostly considered the probability distribution as an extracted feature for classification via means of another classifier such as Bayesian Classifiers, Artificial Neural Networks and Support Vector Machines [6, 16–22].

Such probability models are also prevalent in other fields of study such as natural language processing (NLP), however, they have been deployed in a completely different manner. Instead of considering these models as features for input into other classifiers such as Support Vector Machines (SVM) or k-Nearest Neighbours (kNN), they are considered probabilistic dependency models that determine the likelihood of a protein belonging to a subcellular location. In this research, the linear interpolation smoothing model is proposed which extracts features using syntactical information from the protein sequences for predicting protein subcellular localizations. This approach is a basic approximation technique in NLP and its concepts have been applied in proteomics for this study. Linear interpolation builds probabilistic profiles for proteins based on the frequency information of amino acid subsequences extracted from proteins to perform subcellular localization. These probabilistic profiles may be following the independent or dependent model based on the probabilities being extracted. In this paper, the application of linear interpolation in proteomics is investigated and its ability to predict subcellular localizations of Gram positive and Gram negative bacterial proteins is analyzed.

2 Method

Linear interpolation is *backoff* model, indicating that it aggregates information from different sub-models to determine the likelihood of a protein belonging to a particular class. It builds probabilistic profiles for proteins based on the frequency information of amino acid subsequences extracted from proteins to perform subcellular localization. In this sense, linear interpolation is related to Hidden Markov Models (HMMs) and uses the Markov assumptions to build probabilistic profiles of varying dependencies for proteins, which are later used in this technique to determine the

probability of a protein for belonging to a particular subcellular location.

These probabilistic profiles are similar to amino acid subsequence models that are prevalent in literature, however, their application in linear interpolation is completely different than those previously published. Additionally, there is an absence of techniques, in literature, that aggregate information from various probabilistic models to form a consolidated prediction model. In this scheme, linear interpolation, an approach novel to proteomics, is used to consolidate information from dependent and independent probability distributions to identify the maximum likelihood of a query protein for belonging to a subcellular location.

2.1 Algorithm

Computationally, protein sequences and natural languages share many similarities. They both are ambiguous (similar structures can have different meanings), can be very large, are constantly changing, and they are constructed by a combination of underlying set of constructs, amino acids for protein sequences and words for natural languages. Thus, there is a need to explore the applicability of some basic techniques that are prevalent in NLP for the field of proteomics.

Linear interpolation builds upon probabilistic models of varying dependencies from amino acid subsequences whereby it consolidates the information from these models in an approach known as *backoff*. For clarity, a model of the probability distribution that is dependent on n previous amino acids in the sequence is called the n^{th} probabilistic model (model n, for short). Additionally, it can be noted that Model $n = 0$ is the independent model, since it is not dependent on any other amino acid in the sequence. The probabilistic models studied in this research can be defined as a Markov Chain of order n. With respect to Markov chains of order n, the probability of an amino acid a_i depends only on the immediately n preceding amino acids and not on any other amino acids. In this study, probabilistic models of $n = 0, 1, 2$ are examined as is common with most linear interpolation implementations in NLP.

Mathematically, probability distributions for models $n = 0, 1, 2$ have been defined in Equations 1, 2 and 3 respectively. In the equations, the function *Count()* represents a subroutine that computes the frequency of occurrence of the selected amino acid(s) and *Count(*)* represents the count of all amino acids present in the sequences. a_i represents one of the twenty naturally occurring amino acids in protein samples (this, $i = 1, 2, \ldots, 20$). P denotes the probability of occurrence of an amino acid subsequence for a particular location.

$$P(a_i) = \frac{Count(a_i)}{Count(*)} \tag{1}$$

$$P(a_i|a_{i-1}) = \frac{Count(a_{i-1}, a_i)}{Count(a_{i-1})} \tag{2}$$

$$P(a_i|a_{i-2}, a_{i-1}) = \frac{Count(a_{i-2}, a_{i-1}, a_i)}{Count(a_{i-2}, a_{i-1})} \tag{3}$$

Once the probability distributions have been defined, it is possible to base all predictions from these models individually. In order to compute the probability for a sequence of length N belonging to a particular location, $P(a_{1:N})$, the factoring

the chain rule and then the appropriate Markov assumptions are applied. This has been highlighted in Equations 4, 5 and 6 and they represent the n^{th} probabilistic models with dependencies of *n = 0, 1, 2* respectively.

$$P(a_{1:N}) = \prod_{i=1}^{N} P(a_i) \tag{4}$$

$$P(a_{1:N}) = \prod_{i=2}^{N} P(a_i|a_{i-1}) \tag{5}$$

$$P(a_{1:N}) = \prod_{i=3}^{N} P(a_i|a_{i-2}, a_{i-1}) \tag{6}$$

A major complication of these models is that the probabilities extracted from the training data only provide a rough estimate of the true probability distribution of amino acid subsequences. There may be no representation for uncommon subsequences in the training data, resulting in the probability of 0. This can negatively affect the classification since a zero probability will always yield the resulting overall probability for that class as zero, which may lead to misclassification. Therefore, the model is adjusted so that subsequences with a frequency count of zero are assigned a small non-zero probability and this process of adjusting the probability of low-frequency counts is called smoothing. In this paper, smoothing is done by assigning a probability of 0.0001 to all zero probability subsequences and the probabilities of all other subsequences were adjusted accordingly.

Linear interpolation builds upon these models and aggregates the information present in these models. Subsequence occurrence counts are estimated, but for any sequence that has a low (or zero) count, the model backs off to *n-1* dependency model. In this study, linear interpolation combines the probabilistic models with dependencies of *n = 0, 1, 2* and defines the consolidated probability estimates as:

$$\hat{P}(a_i|a_{i-2}, a_{i-1}) = \lambda_1 \times P(a_i) + \lambda_2 \times P(a_i|a_{i-1}) + \lambda_3 \times P(a_i|a_{i-2}, a_{i-1}) \\ where\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{7}$$

It can be seen from Equation 7 that linear interpolation actually combines probability estimates by weighting the probabilities. λ can be seen as a tuning parameter which determines the overall performance of this model. Similarly, the probability for a sequence of length N belonging to a particular location using linear interpolation can be defined as:

$$P(a_{1:N}) = \prod_{i=3}^{N} \hat{P}(a_i|a_{i-2}, a_{i-1}) \tag{8}$$

From Equations 7 and 8, it can be seen the linear interpolation uses weighted probabilities from the previously discussed probabilistic models and consolidates them to form a unified probabilistic expression. This has been extended using Markov's chain rule to compute the probability of a sequence to belong to a particular subcellular location. However, since the resultant probabilities and their products can be very small, the risk of losing precision in fixed point compute units

is high. Therefore, Equation 8 has been modified to compute the sum of $\log_2$ of the interpolated probabilities as shown below. A similar approach can be applied to the models described previously in Equations 4, 5 and 6 to avoid losing precision on compute units.

$$P(a_{1:N}) = \sum_{i=3}^{N} \log_2 \hat{P}(a_i|a_{i-2}, a_{i-1}) \tag{9}$$

Lastly, it can be seen that if the sequences vary largely in lengths, there arises a need to somehow normalize the resultant probabilities. Normalization can be achieved by dividing the resultant probabilities of the target proteins by their lengths as per this equation:

$$P_{norm}(a_{1:N}) = \frac{P(a_{1:N})}{N} \tag{10}$$

2.2 Optimizing λ

In this scheme, determining the optimal values for λ is key to improving overall performance of linear interpolation because the values for λ determine the weights which are used to aggregate the probabilities from the probabilistic models with dependencies of *n = 0, 1, 2* in linear interpolation and, consequently, alter the probabilities determined for a protein during prediction.

Initially during the early stages of experimentation, equal values for the three λ scalars were chosen for the models, however, this approach does not yield the best results possible using linear interpolation. Moving onwards, the values for λ could be determined using researcher intuition and empirical analysis, however, this approach has its shortcomings such as it is quite slow and requires an extensive manual search, which proceeds by incrementally increasing or decreasing λ until good results are observed. Additionally, this approach does not ensure that optimal values for λ will be discovered. Therefore, a meta-heuristic search and optimization algorithm, the Genetic Algorithm (GA), was chosen to apply optimization techniques to determine the optimal values for λ heuristically. Although this approach has a higher computational cost, the benefits for a better prediction model offsets its costs.

To optimize λ, the chromosomes C, which are the templates for the solution, are evaluated by GA was encoded using real-value between the range $0 \leq C_i \leq 1$, where C_i is the value for a particular gene in the chromosome. The chromosomes had a length $len(C) = 3$ since λ consists of λ_i where *i = 1, 2, 3*. Furthermore, during experimentation it was observed that GA converged fairly quickly during evolution, thus, small values for the generation limit and the population size were needed in order to prevent over training.

In order to evaluate the fitness of every chromosome, the fitness function determines the accuracy during prediction by linear interpolation using the values of λ being processed. The objective of GA was minimization during this experimentation, thus, the fitness function has to return a lower value for the chromosomes that provide better results. There are a number of metrics that can be used to determine the final output of the fitness function. For instance, it is possible to calculate the specificity and/or the sensitivity and return its reciprocal as the fitness value. Since sensitivity and specificity are of equal importance in classification, the fitness

function in this study returned the reciprocal of the means of the sensitivity and specificity values. Additionally, the gene values, C_i, were normalized to determine the values of λ_i prior to the calculation of these metrics. It should be noted that the metrics are calculated over the training samples only using k-fold cross validation during training. The various parameters for GA used during training and other phases of experimentation are listed in Table 1.

Parameter	**Value**
GA Objective	Minimization
Number of Generations	100
Population Size	500
Crossover Rate	0.8
Crossover Function	Two Point Crossover
Mutation Function	Adaptive Feasible
Chromosome Length	3

Table 1: A list of parameters for the Genetic Algorithm

2.3 Overall

In a nutshell, the scheme proposed in this research has been illustrated in Figure 1. Initially, protein sequences are processed to extract the probability profiles for the dependency models of *n = 0, 1 ,2*. Then, linear interpolation is applied to form a consolidated prediction model based on these probabilities. In order to improve performance, the weights used for backoff, λ, are optimized using GA during the training phase as depicted. Once training is completed, the optimized λ values with linear interpolation is used for subcellular localization of target proteins.

3 Results

The performance of the proposed technique has been primarily evaluated using two metrics, sensitivity and specificity. Mathematically, sensitivity and specificity have been described in Equations 11 and 12 respectively shown below. In these equations, *TP* represents the number of samples predicted as positive that belong to the positive class, *FP* represents the number of samples predicted incorrectly as positive that belong to the negative class, *TN* represents the number of samples as negative that belong to the negative class, and *TN* represents the number of samples predicted incorrectly as negative that belong to the positive class.

$$Sensitivity = TP/(TP + FP) \tag{11}$$

$$Specificity = TN/(TN + FN) \tag{12}$$

The performance of probabilistic models with varying dependencies and linear interpolation has been compared and discussed. Although the main concern of this paper is linear interpolation, it was also deemed prudent to show the results

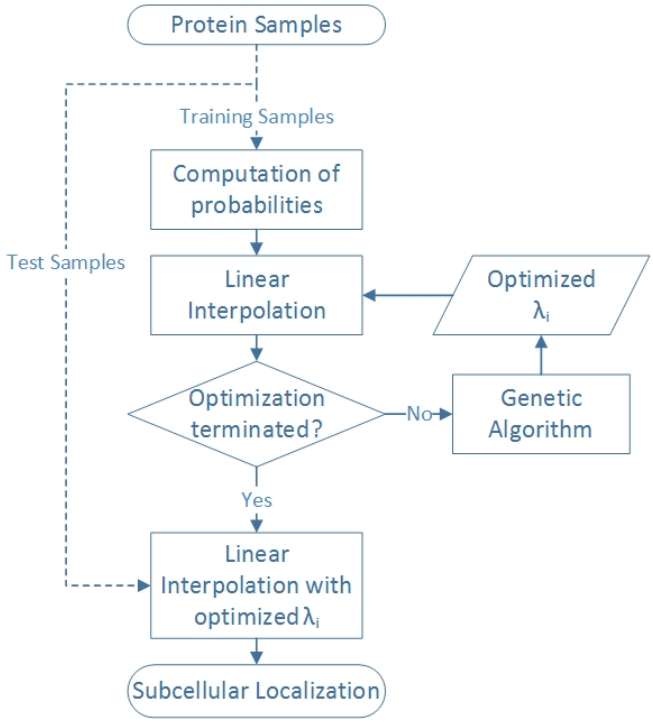

Figure 1: An illustration of the proposed scheme

achieved using the various models for purposes of comparison. Additionally, since λ significantly affects the performance of linear interpolation, a brief comparison of linear interpolation with unoptimized and GA optimized λ values was also necessary.

Authors in previous research have mainly used k-fold cross validation or jackknife tests to report their results. The majority of the experimentation was conducted using the widely accepted k-fold cross validation paradigm for $k = 5, 6, \ldots, 10$. However, for better comparability, jackknife tests were also conducted on linear interpolation. In order to gain statistical stability, the k-fold cross validation was repeated 100 times using random sub-sampling.

The results observed while performing k-fold cross validation have been summarized in Tables 2 and 3 for the Gram positive and Gram negative dataset respectively. In the tables, linear interpolation has been abbreviated as LIP and model n specifies the probabilistic models with dependency n. Since linear interpolation and its underlying models build the protein profiles by computing probabilities of amino acid subsequence occurrences, it is possible to evaluate the impact of adding evolutionary information to the prediction model by the means of computing consensus and using the consensus sequence to perform the computations rather than the raw sequences. For the tables displaying results, a prefix of *PSSM+* indicates that the model builds protein profiles after computing consensus rather than directly using the raw amino acid sequence.

Probabilistic models for dependencies $n = 0, 1, 2$ individually, linear interpolation with equal values for λ ($\lambda_i = \frac{1}{3}$) and liner interpolation with optimized λ values using GA have been compared. Upon referring to the results for the Gram positive dataset displayed in Table 3, it can be noted that linear interpolation with optimized values for λ outperformed other schemes in cross validation for all values

of k.

Since sensitivity and specificity are equally important in any classification task, high values for both metrics indicates a well balanced prediction model. The other models, especially linear interpolation with equal weights, display high values for a metric, however, they have lower performance for the other metric. Additionally, an observation that can be noted is that the techniques which have evolutionary information added via consensus perform slightly better than those without. In the case of linear interpolation, optimizing values of λ instead of using equal values for λ leads to a significant difference in the overall performance.

The results observed for the Gram negative dataset are also similar and linear interpolation with optimized λ performs better than other models. The other models display results that are much closer to those noted for linear interpolation. Similarly, computing features on consensus sequences has slight improvements although they are not as significant as they were in Gram positive dataset.

For a more detailed analysis of the results, the results obtained for each subcellular location in the datasets are reported in Tables 6 and 7 using 10-fold cross validation. These results highlight the previously stated facts that adding evolutionary information via consensus improves performance for the models at every subcellular location in both the datasets. The other models also exhibit good performance, however, linear interpolation with optimized λ is clearly dominant. The distribution of sensitivity and specificity values are also quite balanced indicating that the results are not skewed in either direction due to a disproportionate distribution of these metrics.

For better comparability, the results obtained by performing jackknife tests using linear interpolation with optimized values for λ have been reported in Tables 4 and 5. Jackknife test was not performed on the other models due to its computational costs and also since the main focus of this study is to highlight the applicability of linear interpolation and not its encompassing models. It should be noted that these results are computed after computing consensus on the raw sequences. From the results, it can be seen that the performance of linear interpolation is quite steady and the results obtained from k-fold cross validation and jackknife test are similar. There is high accuracy shown for both the datasets and results for all the locations in the datasets are relatively balanced.

4 Discussion

Our proposed technique builds probabilistic models on primary protein sequences. It utilizes only syntactical and evolutionary information of proteins and, therefore, we gauge the performance of our method with the methods which are mainly based on structural and evolutionary information. This would give a relative measure of performance for the proposed technique when comparing with similar methods.

The results obtained in this study perform on par or better than most of the recently proposed techniques in literature [1, 23–25] for Gram Negative bacterial proteins, however, the results are slightly inferior for Gram Positive bacterical protiens. Since the proposed technique is a learning method that only utilizes syntactical and evolutionary information, we can only compare this strategy with similar work. There are some techniques that have been proposed recently in literature, however, these techniques incorporate functional domains and gene ontology infor-

Scheme		$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Model $n=0$	Sensitivity	70.8	71.7	71.4	71.8	71.7	71.3
	Specificity	83.6	83.8	83.7	83.8	83.6	84.0
Model $n=1$	Sensitivity	73.0	75.4	74.2	74.3	73.4	74.6
	Specificity	81.5	81.4	81.4	81.5	81.2	81.4
Model $n=2$	Sensitivity	74.3	75.4	74.9	74.8	74.8	74.6
	Specificity	82.0	82.2	81.6	82.3	82.3	82.4
LIP	Sensitivity	73.2	73.8	73.0	73.4	73.7	73.7
(Equal λ)	Specificity	83.5	83.6	83.6	83.6	83.4	83.7
LIP	Sensitivity	73.4	73.4	73.9	74.1	74.7	74.9
(Optimized λ)	Specificity	83.3	83.3	82.7	82.9	82.8	82.7
PSSM+Model $n=0$	Sensitivity	75.4	75.4	75.6	75.5	75.4	75.5
	Specificity	83.8	83.9	83.8	84.0	84.0	83.9
PSSM+Model $n=1$	Sensitivity	80.3	80.4	80.4	80.4	80.5	80.2
	Specificity	85.0	84.9	85.0	84.8	85.0	84.9
PSSM+Model $n=2$	Sensitivity	78.8	79.1	79.1	79.4	78.9	79.3
	Specificity	83.5	83.5	83.5	83.6	83.6	83.4
PSSM+LIP	Sensitivity	79.4	80.1	79.9	80.2	80.0	80.4
(Equal λ)	Specificity	84.6	84.5	84.3	84.5	84.5	84.3
PSSM+LIP	Sensitivity	80.2	80.3	80.3	80.4	80.5	80.7
(Optimized λ)	Specificity	84.9	84.9	84.8	84.8	84.8	84.9

Table 2: A summary for the performance of the various models for prediction studied in this paper using the Gram positive bacterial dataset using k-fold cross validation for $k = 5, 6, \ldots, 10$

Scheme		$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Model $n=0$	Sensitivity	80.3	80.0	80.7	80.4	79.6	80.2
	Specificity	84.4	84.3	84.4	84.3	84.3	84.3
Model $n=1$	Sensitivity	84.7	84.5	85.2	85.0	85.1	85.3
	Specificity	77.9	77.7	77.6	77.6	77.7	77.5
Model $n=2$	Sensitivity	86.1	85.2	83.5	85.8	85.8	86.2
	Specificity	72.5	72.3	72.1	72.0	71.9	71.8
LIP	Sensitivity	84.1	84.5	83.6	84.2	83.6	84.8
(Equal λ)	Specificity	79.4	79.3	79.1	79.1	79.1	78.9
LIP	Sensitivity	82.5	80.9	82.1	82.0	81.5	83.0
(Optimized λ)	Specificity	82.1	82.1	82.7	82.3	82.4	81.3
PSSM+Model $n=0$	Sensitivity	79.0	78.7	79.0	78.9	78.9	79.0
	Specificity	86.4	86.4	86.4	86.4	86.4	86.4
PSSM+Model $n=1$	Sensitivity	84.7	84.4	85.0	84.9	85.1	85.0
	Specificity	84.3	84.2	84.1	84.1	84.1	84.1
PSSM+Model $n=2$	Sensitivity	82.0	82.3	82.3	83.1	83.4	83.3
	Specificity	84.9	84.7	84.6	84.5	84.4	84.5
PSSM+LIP	Sensitivity	82.5	82.5	82.7	83.5	83.5	83.5
(Equal λ)	Specificity	86.2	86.1	86.0	85.9	85.9	85.9
PSSM+LIP	Sensitivity	84.8	84.8	84.9	85.3	85.5	85.9
(Optimized λ)	Specificity	86.1	85.9	85.8	85.7	85.7	85.7

Table 3: A summary for the performance of the various models for prediction studied in this paper using the Gram negative bacterial dataset using k-fold cross validation for $k = 5, 6, \ldots, 10$

Subcellular Location	**Accuracy**
Cell membrane	$114/174 = 65.5\%$
Cell wall	$16/18 = 88.9\%$
Cytoplasm	$194/208 = 93.3\%$
Extracellular	$95/123 = 77.2\%$
Overall	$\mathbf{419/523 = 80.1\%}$

Table 4: Results from the jackknife test performed on Gram positive bacterial protein dataset

Subcellular Location	**Accuracy**
Cell inner membrane	474/557 = 85.1%
Cell outer membrane	84/124 = 67.7%
Cytoplasm	375/410 = 91.5%
Extracellular	95/133 = 71.4%
Fimbrium	30/32 = 93.8%
Flagellum	12/12 = 100.0%
Nucleoid	7/8 = 87.5%
Periplasm	137/180 = 76.1%
Overall	**1214/1456 = 83.4%**

Table 5: Results from the jackknife test performed on Gram negative bacterial protein dataset

Scheme		**Locations**				
		Cell membrane	Cell wall	Cytoplasm	Extracellular	**Overall**
Model $n = 0$	Sensitivity	59.9	69.4	80.2	75.6	71.3
	Specificity	87.4	85.2	75.9	87.5	84.0
Model $n = 1$	Sensitivity	57.2	77.8	90.0	73.6	74.6
	Specificity	94.0	74.2	69.5	87.9	81.4
Model $n = 2$	Sensitivity	68.8	75.0	83.9	70.5	74.6
	Specificity	88.3	76.1	80.4	84.9	82.4
LIP	Sensitivity	61.1	73.6	87.0	73.2	73.7
(Equal λ)	Specificity	91.7	79.0	76.4	87.5	83.7
LIP	Sensitivity	67.1	77.8	83.5	71.1	74.9
(Optimized λ)	Specificity	92.1	76.6	74.9	87.3	82.7
PSSM+Model $n = 0$	Sensitivity	68.0	71.4	88.2	74.5	75.5
	Specificity	82.2	87.7	76.8	88.9	83.9
PSSM+Model $n = 1$	Sensitivity	62.8	88.1	92.3	77.8	80.2
	Specificity	91.0	81.6	79.3	87.8	84.9
PSSM+Model $n = 2$	Sensitivity	71.4	84.2	92.0	69.6	79.3
	Specificity	82.2	83.9	79.5	88.2	83.4
PSSM+LIP	Sensitivity	67.9	87.2	92.1	74.3	80.4
(Equal λ)	Specificity	85.8	83.8	79.2	88.6	84.3
PSSM+LIP	Sensitivity	67.2	87.3	91.9	76.3	80.7
(Optimized λ)	Specificity	87.9	83.4	79.8	88.3	84.9

Table 6: A detailed comparison of the various models studied using 10-fold cross validation on Gram positive bacterial dataset

Scheme		**Locations**								
		Cell inner membrane	Cell outer membrane	Cytoplasm	Extracellular	Fimbrium	Flagellum	Nucleoid	Periplasm	**Overall**
Model $n = 0$	Sensitivity	76.3	70.8	90.2	76.5	81.3	93.7	75.0	77.8	80.2
	Specificity	91.3	85.8	74.1	81.9	87.4	93.8	88.8	71.4	84.3
Model $n = 1$	Sensitivity	73.4	87.3	91.8	87.0	87.5	100.0	75.0	80.3	85.3
	Specificity	96.8	69.7	71.6	73.9	82.5	72.7	83.6	69.4	77.5
Model $n = 2$	Sensitivity	79.4	84.3	87.2	89.8	96.9	100.0	81.3	70.6	86.2
	Specificity	91.3	71.6	76.2	69.2	63.2	54.1	68.5	80.6	71.8
LIP	Sensitivity	76.3	81.3	90.3	86.8	90.6	100.0	78.1	75.0	84.8
(Equal λ)	Specificity	94.5	76.0	74.9	73.5	79.6	77.6	79.5	75.9	78.9
LIP	Sensitivity	76.5	75.6	90.1	81.6	85.9	100.0	78.1	76.5	83.0
(Optimized λ)	Specificity	92.7	81.1	74.7	77.7	82.6	84.0	84.1	73.4	81.3
PSSM+Model $n = 0$	Sensitivity	79.7	64.5	90.4	71.9	81.3	100.0	65.0	78.9	79.0
	Specificity	88.5	90.4	74.7	87.3	91.8	97.2	90.5	71.0	86.4
PSSM+Model $n = 1$	Sensitivity	77.6	81.2	91.5	80.6	88.8	100.0	85.0	75.6	85.0
	Specificity	93.6	79.2	76.9	84.9	89.8	90.0	81.1	77.0	84.1
PSSM+Model $n = 2$	Sensitivity	85.0	67.6	91.1	71.6	95.0	100.0	81.3	75.3	83.3
	Specificity	85.2	86.9	80.1	86.8	86.9	89.5	80.1	80.1	84.5
PSSM+LIP	Sensitivity	81.7	68.7	91.5	74.5	90.6	100.0	83.1	77.4	83.5
(Equal λ)	Specificity	89.9	86.7	78.5	86.8	89.4	93.5	84.6	77.9	85.9
PSSM+LIP	Sensitivity	81.8	81.0	91.3	77.3	90.5	100.0	83.6	76.9	85.3
(Optimized λ)	Specificity	89.4	86.9	78.3	86.7	89.5	93.1	84.7	77.2	85.7

Table 7: A detailed comparison of the various models for prediction studied using 10-fold cross validation using the Gram negative bacterial dataset

mation [26, 27]. It is in general time consuming for newly extracted proteins to annotate and record in such a large database, therefore, it may not be possible to use such techniques for predicting the subcellular localization of these proteins.

Nonetheless, incorporating functional information and gene ontology information will significantly improve the performance. The proposed technique builds probabilistic models on the primary protein structure only, therefore, does not rely on functional information. A comparison of reported sensitivity and specificity values for Gram Positive and Gram Negative datasets that have been recently published are shown in Table 8.

Scheme	**Reported Results**			
	Gram Positive		**Gram Negative**	
	5-fold	10-fold	5-fold	10-fold
Pacharawongsakda 2013 [23]	-	-	-	73.2%
Huang 2013 [1]	80.4%	-	-	-
Dehzangi 2014 [25]	-	83.6%	-	76.6%
This paper	**80.2%**	**80.7%**	**84.8%**	**85.9%**

Table 8: A comparison of recently published results for Gram Positive and Gram Negative datasets.

Although linear interpolation displays reasonable results, there is scope for further improvement. This paper is aimed at introducing the possibility of applying a basic natural language processing technique in the field of proteomics. There are numerous possibilities that can be explored to improve the performance of linear interpolation, which have been highlighted in the *Conclusion* section.

There is a major distinction between the learning technique explored in this study and several of the other classification techniques published previously in the literature. Linear interpolation can be categorized as a maximum likelihood technique since it predicts the class of a sample based on the computed probabilities. In essence, it determines class labels by the highest computed probability. This allows linear interpolation to be quite robust and modular, and it relatively easier to extend this technique without significantly increasing the computational cost. For instance, in this study, probabilistic models with dependencies (of up to $n = 2$) have been discussed, however, the technique can be easily modified to include higher order dependency models to profile proteins.

Although there is an additional computation involved in computing the frequencies of the amino acid subsequences, the prediction process itself does not experience any drastic increases in computational cost since its simply the maximum of cumulative sums of probabilities of the various dependency models. Conventional classifiers (like SVM) have drastic affect in performance when the dimensionality is increased (which increases the computational cost). However, linear interpolation is able to deal with high dimensionality problems since after forming the dependency models, classification occurs by simply suming up the probabilities of these models and selecting the class with the greatest probability. For instance, dependency model $n = 0$ has 20 unique probabilities per class, $n = 1$ has 400 unique probabilities per class and $n = 2$ has 8000 unique probabilities per class. This stage, when computing the probabilities, can be seen as a feature extraction task, which has a computational

complexity of $O(20^{n+1})$. However, when computing the likelihood of a query sample (the classification stage) belonging to a particular subcellular location, every model computes a sum that represents the overall probability of that sample belonging to that particular location, which has the computational complexity of $O(n)$.

5 Conclusion

It has been shown in this work that there is significant potential for linear interpolation in protein subcellular localization. The proposed method has shown reasonable results for both Gram Positive and Gram Negative bacterial proteins. Currently, we are working on providing the revelant code as part of an open source library for public use.

Furthermore, it may be possible to further improve the results if optimization of λ is done using some other optimization techniques. Currently, GA allows for global optimization, however, it is difficult to fine-tune or find the local optima in the global search space using GA. However, if a local optimization algorithm with GA is used, such as simulated annealing or even artificial neural networks, the performance of the proposed technique could be improved.

Additionally, since the dimensionality of this approach is "independent" of the depth of underlying dependency models at the classification stage, it is possible to explore the effects of increasing the order of dependencies n. The computational cost of increasing n can be offset by an increase in the classifier performance.

Lastly, the discussed technique builds the dependency models using amino acid occurrence frequencies from either the raw primary sequences or after taking consensus. However, these dependency models can be built directly from the information present in PSSM, and this approach can be explored to investigate if it leads to any improvements in prediction.

References

[1] Chao Huang and Jingqi Yuan. Using radial basis function on the general form of Chou's pseudo amino acid composition and PSSM to predict subcellular locations of proteins with both single and multiple sites. *Biosystems*, 2013.

[2] Shengnan Tang, Tonghua Li, Peisheng Cong, Wenwei Xiong, Zhiheng Wang, and Jiangming Sun. PlantLoc: an accurate web server for predicting plant protein subcellular localization by substantiality motif. *Nucleic acids research*, 2013.

[3] Suyu Mei, Wang Fei, and Shuigeng Zhou. Gene ontology based transfer learning for protein subcellular localization. *BMC bioinformatics*, 12(1):44, 2011.

[4] E Tantoso and Kuo-Bin Li. AAIndexLoc: predicting subcellular localization of proteins based on a new representation of sequences using amino acid indices. *Amino Acids*, 35(2):345–353, 2008.

[5] Tanwir Habib, Chaoyang Zhang, Jack Y Yang, Mary Qu Yang, and Youping Deng. Supervised learning method for the prediction of subcellular localization of proteins using amino acid and amino acid pair composition. *BMC genomics*, 9(Suppl 1):S16, 2008.

[6] Annette Höglund, Pierre Dönnes, Torsten Blum, Hans-Werner Adolph, and Oliver Kohlbacher. MultiLoc: prediction of protein subcellular localization using N-terminal targeting sequences, sequence motifs and amino acid composition. *Bioinformatics*, 22(10):1158–1165, 2006.

[7] Kuo-Chen Chou. Some remarks on protein attribute prediction and pseudo amino acid composition. *Journal of theoretical biology*, 273(1):236–247, 2011.

[8] Xuan Xiao, Zhi-Cheng Wu, and Kuo-Chen Chou. A multi-label classifier for predicting the subcellular localization of gram-negative bacterial proteins with both single and multiple sites. *PLoS One*, 6(6):e20592, 2011.

[9] Hong-Bin Shen and Kuo-Chen Chou. Virus-mPLoc: a fusion classifier for viral protein subcellular location prediction by incorporating multiple sites. *Journal of Biomolecular Structure and Dynamics*, 28(2):175–186, 2010.

[10] Hong-Bin Shen and Kuo-Chen Chou. Gneg-mPLoc: a top-down strategy to enhance the quality of predicting subcellular localization of Gram-negative bacterial proteins. *Journal of Theoretical Biology*, 264(2):326–333, 2010.

[11] Kuo-Chen Chou and Hong-Bin Shen. Plant-mPLoc: a top-down strategy to augment the power for predicting plant protein subcellular localization. *PloS one*, 5(6):e11335, 2010.

[12] Kuo-Chen Chou, Zhi-Cheng Wu, and Xuan Xiao. iLoc-Hum: using the accumulation-label scale to predict subcellular locations of human proteins with both single and multiple sites. *Molecular Biosystems*, 8(2):629–641, 2012.

[13] Xiaomei Li, Xindong Wu, and Gongqing Wu. Robust feature generation for protein subchloroplast location prediction with a weighted GO transfer model. *Journal of theoretical biology*, 347:84–94, 2014.

[14] Pufeng Du and Yanda Li. Prediction of protein submitochondria locations by hybridizing pseudo-amino acid composition with various physicochemical features of segmented sequence. *BMC bioinformatics*, 7(1):518, 2006.

[15] Sebastian Briesemeister, Jörg Rahnenführer, and Oliver Kohlbacher. Going from where to whyâĂŤinterpretable prediction of protein subcellular localization. *Bioinformatics*, 26(9):1232–1238, 2010.

[16] C H Q Ding and Inna Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[17] Pradip Ghanty and Nikhil R Pal. Prediction of protein folds: Extraction of new features, dimensionality reduction, and fusion of heterogeneous classifiers. *NanoBioscience, IEEE Transactions on*, 8(1):100–110, 2009.

[18] Alok Sharma, James Lyons, Abdollah Dehzangi, and Kuldip K Paliwal. A feature extraction technique using bi-gram probabilities of position specific scoring matrix for protein fold recognition. *Journal of theoretical biology*, 320:41–46, 2013.

[19] Harsh Saini, Gaurav Raicar, Alok Sharma, Sunil Lal, Abdollah Dehzangi, Rajeshkannan Ananthanarayanan, James Lyons, Neela Biswas, and Kuldip K Paliwal. Protein Structural Class Prediction via k-separated bigrams using Position Specific Scoring Matrix. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 18(4):474–479, 2014.

[20] Kuldip K Paliwal, Alok Sharma, James Lyons, and Abdollah Dehzangi. A Tri-Gram Based Feature Extraction Technique Using Linear Probabilities of Position Specific Scoring Matrix for Protein Fold Recognition. *NanoBioscience, IEEE Transactions on*, 13(1):44–50, 2014.

[21] Harsh Saini, Gaurav Raicar, Sunil Lal, Abdollah Dehzangi, James Lyons, Kuldip K Paliwal, Seiya Imoto, Satoru Miyano, and Alok Sharma. Genetic algorithm for an optimized weighted voting scheme incorporating k-separated bigram transition probabilities to improve protein fold recognition. In *2014 Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, pages 1–7. IEEE, 2014.

[22] Alok Sharma, Kuldip K Paliwal, Abdollah Dehzangi, James Lyons, Seiya Imoto, and Satoru Miyano. A strategy to select suitable physicochemical attributes of amino acids for protein fold recognition. *BMC bioinformatics*, 14(1):233, 2013.

[23] Eakasit Pacharawongsakda and Thanaruk Theeramunkong. Predict Subcellular Locations of Singleplex and Multiplex Proteins by Semi-Supervised Learning and Dimension-Reducing General Mode of Chou's PseAAC. *IEEE Transactions on Nanobioscience*, 2013.

[24] Kuo-Chen Chou and Hong-Bin Shen. Cell-PLoc: a package of Web servers for predicting subcellular localization of proteins in various organisms. *Nature protocols*, 3(2):153–162, 2008.

[25] Abdollah Dehzangi, Rhys Heffernan, James Lyons, Alok Sharma, Kuldip Paliwal, and Abdul Sattar. Gram-Positive and Gram-Negative Subcellular Localization Using Rotation Forest and Physicochemical-Based Features. In *Pattern Recognition in Bioinformatics: 9th IAPR International Conference, PRIB 2014, Stockholm, Sweden, August 21-23, 2014. Proceedings*, volume 8626, page 112. Springer, 2014.

[26] Kuo-Chen Chou and Hong-Bin Shen. Cell-PLoc 2.0: An improved package of web-servers for predicting subcellular localization of proteins in various organisms. *Natural Science*, 109:1091, 2010.

[27] Xuan Xiao, Zhi-Cheng Wu, and Kuo-Chen Chou. iLoc-Virus: A multi-label learning classifier for identifying the subcellular localization of virus proteins with both single and multiple sites. *Journal of Theoretical Biology*, 284(1):42–51, 2011.