Boon Thye Thomas Yeo
4662474
Biochem 218 Final Project

Using Information Theory to Reduce Complexities of Neural Networks in Protein Secondary Structure Prediction

1 Abstract

Neural networks are commonly used in protein secondary structure prediction. However, neural networks tend to be very computationally expensive and time consuming because the training algorithms have to determine the values of thousands of variables that define the networks. In this paper, we suggest a simple information theoretic method of pre-processing the training set before feeding it to a generic neural network, which should theoretically reduce the complexities of the neural network significantly, thus saving valuable computational time. Due to insufficient time, we did not actually physically implement a neural network with a pre-processing stage. Instead, we shall only demonstrate the feasibility of our suggestion by pre-processing the family of leucine zippers and show that it achieves the aim of obtaining improved inputs to the network. Hopefully, our future work will involve implementing such a neural network with a pre-processing stage to determine the actual improvement in computational time.

2 Introduction to Neural Networks and Structure Prediction

The three-dimensional structure of a protein is very important to the function of a protein. Changes in the shape of active sites (due to residue mutation) can result in functional failure of the protein. Knowing the three-dimensional structure of a protein can help in the design of agents (such as drugs) that target the protein. However, physical simulations have not been successful in determining the tertiary structures from the primary sequences (Rost & Sander, 1993). The prediction problem is simplified if we reduce the three-dimensional problem to one dimension by assigning to each residue a secondary structure (Rost & Sander, 1993): helix (α), strand (β) or loop (L).

The most successful secondary structure prediction methods exploit evolutionary information from multiple alignment of sequences of protein families (Cuff & Barton, 2000). Of the different prediction methods, neural network algorithms are found to be the most accurate (Cuff & Barton, 2000). For example, the neural network algorithm developed by

Rost & Sander (1993) achieved an accuracy of above 70%[1]. Since then, various improvements (such as the careful use of iterated PSI-BLAST) and circumstances (such as larger sequence databases) have resulted in a rise of accuracy to around 80% (Cuff & Barton, 2000). Yet, the fundamentals of neural net prediction algorithms have not really been changed. The first step still involves creating a training set of proteins that have been multiple aligned[2] and whose secondary structures are known. The neural network is then trained with this training set. The accuracy of the trained network is then measured with a test set.

We shall now give a quick outline of the neural network algorithm Rost & Sander used in their 1993 paper[3]. The neural network utilized has 3 levels as shown in figure 1. The training set for the network is first multiple aligned. For each residue, the frequency of occurrence, $f_i,\ 1 \leq i \leq 20$, of each of the 20 amino acids at that position is calculated. Since we are interested in determining the secondary structure of each residue, each residue should have its own Threshold Logic Unit (TLU) in the first level (figure 1). Consider a window of $w = 13$ consecutive residues, with the center residue being the one whose secondary structure we are interested in. The input to the TLU is an input vector of length, $w = 13$ ($w = 7$ in figure 1), consisting of a data vector from each of the residues in the window. The i^{th} element of each data vector is f_i[4]. Hence each TLU takes in $w \times 21$ real numbers. The TLU has 3 output components corresponding to the secondary structures helix (α), strand (β) and loop (L). If the central residue has a helical structure, then our desired output is $(1, 0, 0)$ (although what we typically get is something like (0.54, 0.21, 0.25)). For the j^{th} output component, the TLU takes the $w \times 21$ real numbers, summed them together in a linear, weighted fashion to give the real number, h_j. The i^{th} output component is then given by the sigmoid function, $f_j(h_j) = \frac{1}{1+e^{-\beta_j h_j}}$, where β_j is the slope of the sigmoid function. Hence, for this TLU alone, we need to determine the values of $w \times 21 \times 3$ weights and $3\beta_j's$.

$L = 17$ consecutive TLUs (figure 1 shows 5) now pipe their outputs to a basic cell at the second level. Once again, the TLU of the residue of interest is at the center of the input vector to the basic cell. The second level operates like the first except that this time round, there are $L \times 3 \times 3$ weights (because there are $L \times 3$ inputs and 3 outputs) and $3\beta's$ to be determined for each basic cell.

The gradient descent method is used to determine the variables in both level 1 and level 2. Basically the aim is to minimize the error of the output. Remembering that the

[1]The different definitions of accuracy of protein structure prediction is well-elaborated in Ross & Sander, 1993. A widely accepted measure of accuracy is the three-state per-residue accuracy, Q (Cuff & Barton, 2000) also discussed in the same 1993 paper by Ross & Sander.

[2]While this set of proteins should belong to the same family, there should be as much divergence within the family as possible (Przybylski & Rost, 2002). This expands the predictive power of the neural network by preventing an overfit of the training set.

[3]http://gnosis.cx/publish/programming/neural_networks.htm provides an excellent introduction to neural networks if the reader is not familiar with their use.

[4]While there are 20 amino acids, there is actually a f_{21} because we need to create a dummy amino acid for residues at the terminals of the protein chains without neighbors on both sides. Refer to Rost and Sander, 1993, for a more detailed discussion.

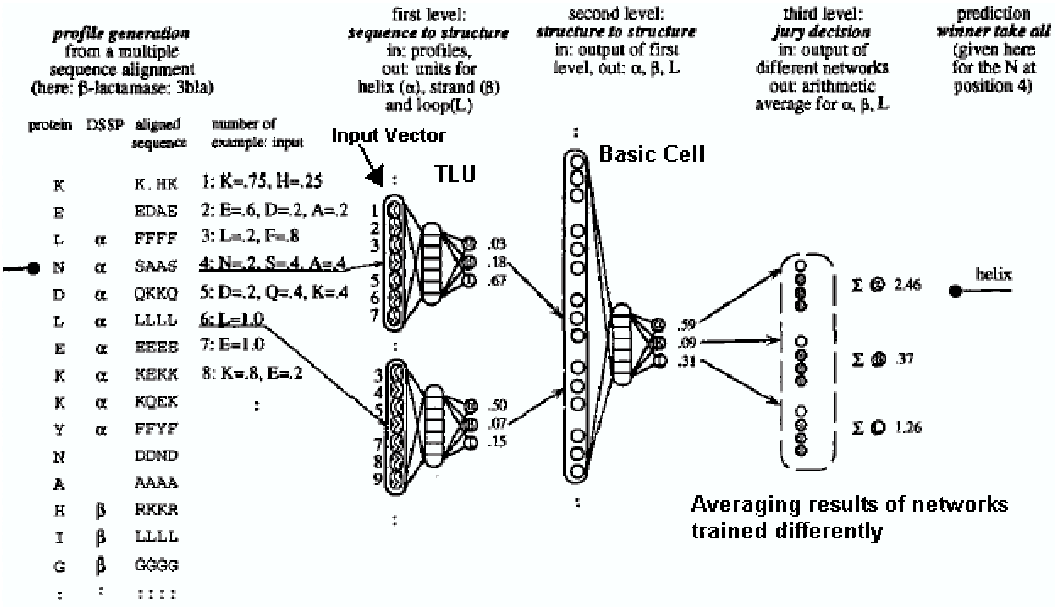

Figure 1: Neural Network used in Ross & Sander, 1993

output is a $\Re^3$ vector, we define the error to be the sum of the square of difference between each of the actual and desired output components. Since we have so many variables to adjust, we can achieve an error of zero, but that would result in over-fitting. Hence, training is considered complete when 70% accuracy is achieved for the training set for the first level and 75% for the second level.

Finally, Rost & Sander varied the way levels 1 and 2 are trained, thus producing 8 or 9 different set of level 1 and 2 architectures with different junction weights and sigmoid functions. The third level then combines the results of these architectures by averaging the outputs of the second levels of the different architectures (figure 1 only show the averaging of 4 architectures at the third level).

3 Data and Analysis

3.1 Reducing Complexities with Mutual Information

It is clear that training the neural network requires the determination of thousands of variables, which is very computationally expensive. We can reduce the complexities of the neural network by improving the input vector to each TLU in the first level and each basic cell in the second level. By improving the input vector, we can decrease the size of the window used (in both first and second level) and still maintain accuracy.

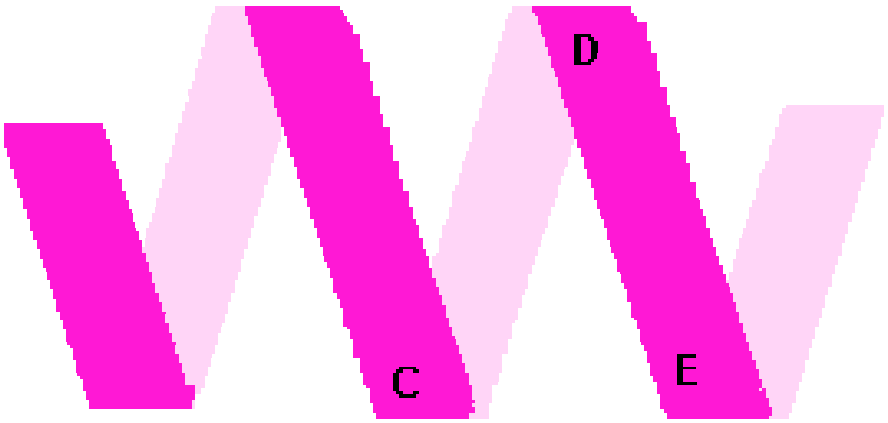

Figure 2: A Representation of an alpha helix. As can be seen, while amino acid D is closer to C in the primary sequence, in 3-dimensional space, E is the one closer to C. Hence, we would expect amino acid E to have stronger statistical correlation with C.

The problem with the implementation outlined in the previous section is that the selection of the elements of the input vector is too arbitrary. The current method assumes that the secondary structure of a residue is most affected by the amino acids closest to it in the *primary sequence*. But, this is not necessarily true, because we are talking about an entity in 3-dimensional space, not a 1-dimensional line. Therefore, the amino acids that affect the secondary structure of a residue the most, should be those closest to it in *3-dimensional space*. It would seem then that we have a catch-22 situation because knowing the amino acids closest to a residue would appear to require us to first know the 3-dimensional structure of the protein.

However, we should be able to make use of the fact that the 3-dimensional structure of a protein is important to the function of a protein. Changes in the structure of a protein can render it non-functional in an organism, thus such mutation can be potentially fatal. A mutation of a residue should affect the residues close to it in 3-dimensional space. If the mutation happens to be a bad one, we should expect that the organism would not survive unless there is a corresponding change in some close surrounding residues to mitigate the bad mutation. Hence we should expect some sort of statistical correlation among residues close together in 3-dimensional space (see figure 2 for illustration).

The use of the concept of mutual information can help us detect and quantify this correlation. We can then let the input vector to the TLU be composed primarily of residues with high correlation to the residue of interest. The input to the basic cell of the second level should also come from TLUs of the highly correlated residues. This ensures that the

inputs have the most effects on the outputs of the TLU and basic cell of the residue of interest (something Rost & Sander's implementation of the neural network algorithm does not guarantee). Therefore, it seems likely that smaller size windows will still achieve similar accuracy. This would imply less variables resulting in a shorter runtime.

To verify and quantify this improvement, we would need to implement a neural network with the pre-processing stage to pick the window. But unfortunately, we lack the time to implement it. Instead, we will investigate the feasibility of the use of mutual information to detect such correlations. An important question that arises is :"Are the correlations actually significant and detectable?".

While strong correlations have been shown to exist in various RNA and DNA studies[5] between nucleotides, and are detectable by mutual informatic methods, we have not found papers that support that similar methods would work for amino acids. Such correlations are almost obvious, for example in tRNA studies, where adenine(A) tends to form hydrogen bonds only with uracil(U) and guanine(G) only bind with cytosine(C). Hence in the study of the cloverleaf secondary structure of tRNA, we would expect that Watson-Crick base pairs are highly correlated. A mutation in one nucleotide will only likely be successful if and only if there is an appropriate mutation in the corresponding nucleotide (if one exists). This has been experimentally verified in the project by Segars (Segars, 2001). For proteins however, there is no "corresponding amino acid". In many cases, the 3-dimensional structure of a protein is not severely affected by amino acid mutation. A small hydrophobic amino acid should be replaceable by another small hydrophobic amino acid without appreciable effect on the 3-dimensional or secondary structure of the protein (this is reflected by substitution matrices such as Blosum or Pam). We might then expect that correlations between nearby residues to be significantly less than that in the case of RNA or DNA. Would these correlations then be detectable? From our experiment on the family of leucine zippers, it turns out that correlations due to secondary structural effects are very strong and detectable. We picked the family of leucine zippers because the family is already self-aligned because of the periodic nature of the sequences, and because the secondary structure of the leucine zipper family is already known (alpha helical structure).

3.2 A Brief Discussion of Mutual Information

Mutual information is a measure of the amount of information one random variable contains about another random variable. It is the reduction in the uncertainty of one random variable due to the knowledge of the other (Cover & Thomas, 1991). Mutual information between 2 random variables is defined to be

$$I(X;Y) = \sum_{x \in A} \sum_{y \in B} \; p(x,y) log_2 \frac{p(x,y)}{p(x)p(y)}$$

[5]For example, refer to the work done by Tom Schneider, or refer to the Biochem 218 Final Project by Paul Segars, 2001 (see References).

Mutual information is measured in bits. Since we are trying to correlate two positions of a family of proteins, X (or Y) would be the random variable whose value is that of the amino acid at position u (or v) after multiple alignment (we can number the amino acids from 1 to 20). A and B refer to the set of values that can be assumed by the random variables X and Y respectively. Thus $A = B = \{1, 2...20, 21\}$, where 21 refers to a gap in the alignment, while 1 to 20 refer to each of the amino acids.

There are a number of important facts we should take note of:

(1) $I(X;Y) \geq 0$, and equality holds if and only if X and Y are independent.

(2) Mutual information is symmetric, that is $I(X;Y) = I(Y;X)$.

(3) $I(X;Y) > I(U;V)$ implies that X gives more information about Y than U about V.

(4) Consider the random variables, $X = 0$ or 1 with equal probability, and $Y = X$. Also, consider the random variables $U = 0$ with probability 1, and $V = U$. In both cases, knowing X determines Y, and knowing U determines V. However, $I(X;Y) = 1$ bit, while $I(U;V) = 0$ bit. This is because to begin with, there is more uncertainty about Y than V. We know that V has to take the value of 1, but we only know that Y has to be the same as X, but it could be 0 or 1 with equal probability. Hence, X reduces the uncertainty of Y more than U reduces the uncertainty of V, even though Y and V are totally dependent on X and U respectively. Thus, we conclude that mutual information is not a measure of degree of *dependency* between two variables (else $I(X;Y)$ should equal $I(U;V)$), but rather how much *information* one variable provides about another.

(5)Obviously, we do not know the actual marginal and joint probability distribution, but we can estimate $p(x, y)$ and $p(x)$ of a family of proteins empirically by the following equations:

$$p(x,y) \sim \hat{p}(x,y) = \frac{\#\ of\ sequences\ with\ amino\ acid\ x\ at\ position\ u\ and\ amino\ acid\ y\ at\ position\ v}{total\ \#\ of\ sequences}$$

and

$$p(x) \sim \hat{p}(x) = \frac{\#\ of\ sequences\ with\ amino\ acid\ x\ at\ position\ u}{total\ \#\ of\ sequences}$$

(6)Lastly, even though two positions X and Y might be independent and $I(X;Y) = 0$, since we have to estimate the joint and marginal distribution, it is therefore not likely the estimated mutual information is going to be zero. This is especially so in our case as we have less than 100 sequences to analyze.

3.3 Data and Methods

For our experimentation, we obtained the family of leucine zippers from Swiss-Prot, because of its reliability (we also tried the Protein Data Bank (PDB) but did not find many leucine zippers). The leucine zipper sequences are known to have a regular repetition of leucine at every 7^{th} position (although in some cases, we found that leucine had been replaced with isoleucine or even tyrosine), and have an alpha helical structure. We found that about half the leucine zippers had 4 leucines spaced apart (22 amino acids long), while the other half had 5 such leucines (29 amino acids long). Because we would have a very small family if we only choose either group, we simply throw away the last 7 amino acids of the group of leucine zippers with 5 equally-spaced leucines. We also threw away sequences of odd length because of deletions (such as 21 amino acids long), as we were unable to align them properly. There were only about 4 of these, so it did not adversely affect the size of our family. In total, we have 62 leucine zippers (see Appendix 1 for the amino acid sequences of the family of leucine zippers).

With the family self-aligned, we then estimated the marginal probability distribution of amino acids of each position (all 22 of them), as well as the joint probability distribution of amino acids for every pair of positions. We then produced the 22×22 matrix, M, whose (i, j) entry is the mutual information between position i and position j. Since mutual information is symmetrical, we only had to fill in the upper triangular portion of M.

Because mutual information does not directly gives us the degree of dependency between 2 positions of a sequence, but also takes into account the marginal distribution of each position (refer to fact #4 in previous subsection), it means that if a pair of positions has a higher $I(X;Y)$ than $I(U;V)$, that does not necessarily imply that Y is more dependent on X than V is on U. Thus, we should estimate the randomized mutual information, $I(X;Y)$, given that X and Y are independent, and subtract this "background noise" from our experimental mutual information M. We had already previously estimated the marginal amino acid probability distribution of each of the 22 positions. We now randomly generated 1000 sets of 62 proteins, each of length 22, such that the amino acid at each position was generated independently using the estimated marginal probability distribution for that position. For each set of proteins, we calculated the 2-dimensional mutual information matrix, N_i. Thus N is a 3-dimensional matrix with dimensions $22 \times 22 \times 1000$. Using N we produced the 22×22 matrices, EN (and $stdN$), whose (i, j) entries are the mean (and standard deviation) of randomized mutual information between position i and j assuming the positions are independent. We then produced the normalized mutual information matrix ($NormM$), whose (i, j) entry is given by:

$$NormM(i,j) = \frac{M(i,j) - EN(i,j)}{stdN(i,j)}$$

Thus $NormM(i, j)$ tells us how significant $M(i, j)$ actually is. Given $M(i, j) = 3$ bits, we

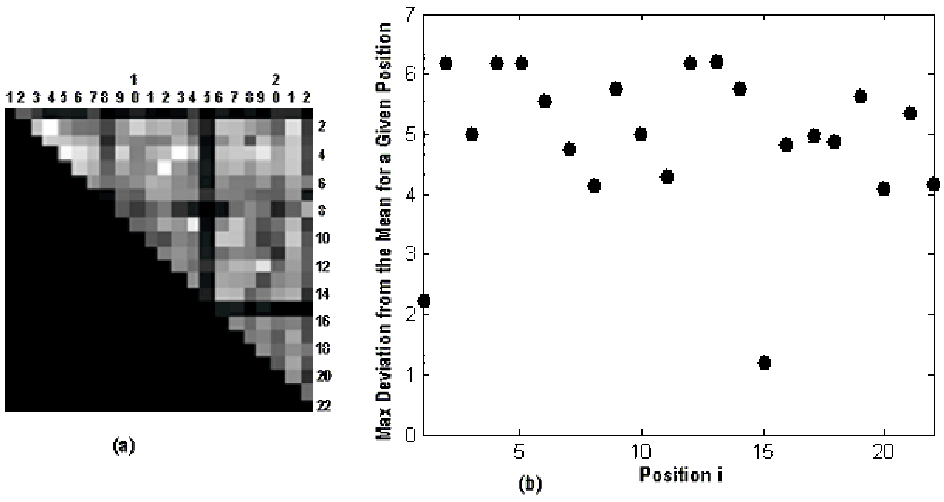

Figure 3: (a)Gray Scale Image of Normalized Mutual Information. (b)Plot of the Maximum Deviation from the Mean for Position i.

have no idea whether this is significant, or whether this can occur by random chance when i and j are independent (see fact #6 from previous subsection).

3.4 Results and Discussion

The matrix $NormM$ is shown in Appendix 2. In figure 3a, we display $NormM$ as a gray-scale image. For values less than or equal to zero, we set the color to black. Bright pixels correspond to positions with significant correlation. Indeed, if we define $maxI_i = max\{NormM(i,j), \forall j \neq i\}$ (which is the maximum normalized correlation obtained for a position i), then $maxI_i$ are all at least 4 deviations above the mean except when $i = 1$ and $i = 15$ (see figure 3b).

Since the leucine zipper family has a periodic structure, alpha helix (refer to figure 2), this periodicity should be included in the normalized mutual information matrix, if the matrix really does reflect the 3-dimensional structure of the protein. Given a residue at position i, and $j > i$, as we increase j, the distance in 3-dimensional space between residues i and j should increase then decrease, then increase and then decrease, and so on. Hence ideally we hope to see some sort of a decaying sinusoidal with a period of about 7 if we make a plot of normalized mutual information against j. In figure 4, we made a plot of normalized mutual information between position 1 and position j, $2 \leq j \leq 22$ and between 2 and position j, $3 \leq j \leq 22$ (basically, the horizontal, non-zero entries of $NormM$). In the top graph, we see a peak at 7, 14, and 20. In the bottom graph, we see a peak at 10, 16 and 21. Thus, from the plots, we notice that there exists some sort of cyclical process,

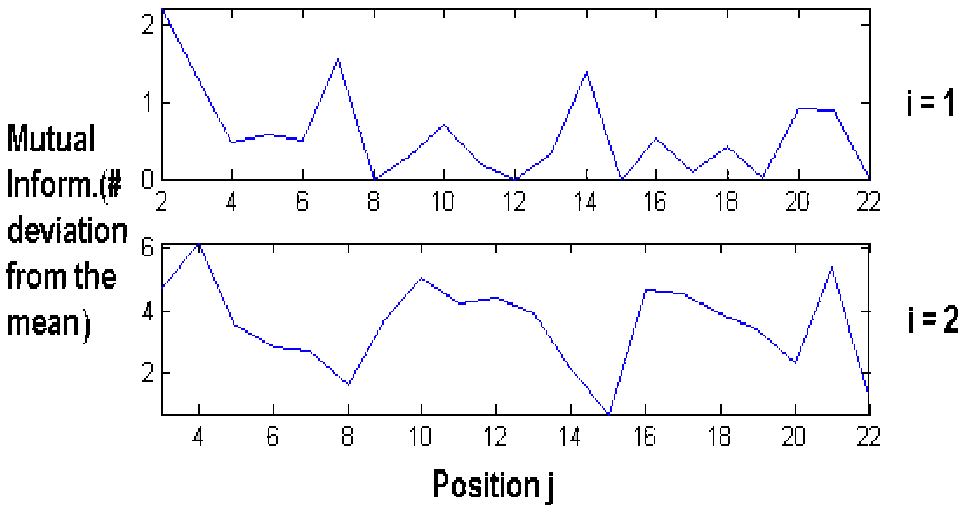

Figure 4: Top: Graph of normalized mutual information between residues 1 and j, $2 \leq j \leq 22$. Bottom: Graph of normalized mutual information between residues 2 and j, $3 \leq j \leq 22$

whose period varies roughly from 5 to 7. Notice also that the peaks appear to be decaying. However, if we observe the rows of $NormM$(Appendix 2), not all the rows appear to conform to the image of a decaying sinusoid.

Since not every row clearly exhibit a period of around 7, it is hard to determine whether or not periodicity is reflected in the normalized mutual information matrix. If it is noise that is hiding the periodicity, we should be able to get rid of the noise by doing some sort of averaging. We first formed the matrix $P' = NormM + NormM^T$, where $NormM^T$ is the transpose of $NormM$. Thus $P'(i,j)$ is the normalized mutual information between position i and position j of the leucine zipper family. Note that the diagonal entries of P' is zero. We then derived the 22×21 matrix, P from P', by removing the diagonal of zeroes. Thus, if $j < i$, $P(i,j)$ is the normalized mutual information between position i and the $(i-j)^{th}$ residue left of position i, and if $j \geq i$, $P(i,j)$ is the normalized mutual information between position i and the $(j-i+1)^{th}$ residue right of position i[6].

Since row i of matrix P represents normalized mutual information between position i and other positions in the sequence, it does not really make sense to simply add the rows of the matrix P, because the j^{th} column is not the normalized mutual information between residues that are spaced equally apart. However, we do want to do some form of averaging. Therefore we decided to use the concept of the discrete fourier transform (DFT). For an input vector $\vec{x}$ of length N, $DFT(\vec{x}) = \hat{x}$ is also a vector of length N,where:

[6]The reader might want to draw out the matrix P' and P in order to visualize the situation when the diagonal of zeros is removed.

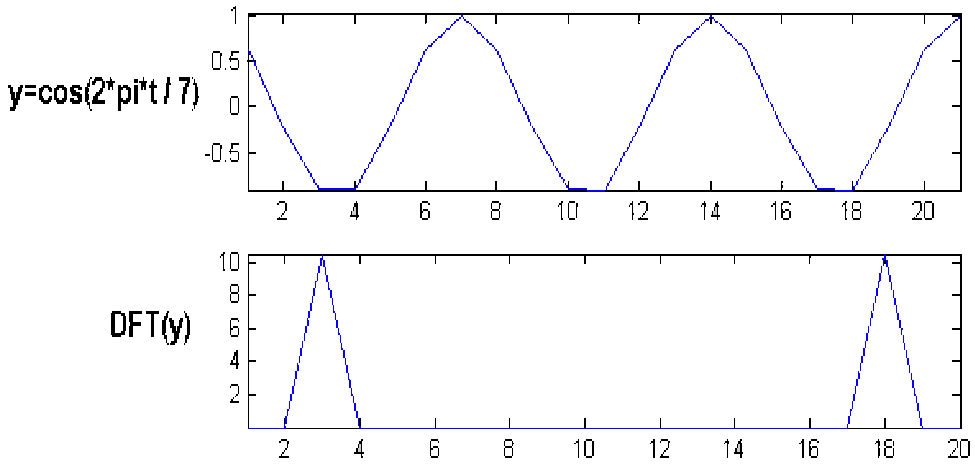

Figure 5: Top: Graph of function $y = cos(2\pi t/7)$, sampled at integer values of t. Note that y has a period of 7. Bottom: DFT of y excluding the first element of the DFT (the first element simply reflects the average value of y). Hence a period of 7 is shown up as a peak at the third point (excluding the first point) of a 22-points DFT.

$$\hat{x}(n) = \sum_{k=1}^{N} \vec{x}(k)\ e^{\frac{-2\pi jnk}{N}}$$

Take note of the following facts:

(1)In our case, while $\vec{x}$ is a real vector, $\hat{x}$ is likely to be complex.

(2)For a real vector $\vec{x}$, only the first half of $\hat{x}$ gives information about the periodicity of $\vec{x}$. The second half of $\hat{x}$ is simply the complex conjugate of the first half. Thus if we plot the graph of the magnitude of $\hat{x}$, it will be symmetrical about the half-way point.

The DFT of a vector detects periodicity in the vector[7]. This is illustrated in Figure 5. The top graph is the sampled function of the cosine function with a period of 7. The DFT of this periodic function is then plotted on the bottom graph (excluding the first or average term of $\hat{x}$). A visible peak is seen at the third point of the DFT. Notice that the graph is symmetrical about the half-way point as mentioned previously.

We now return to the analysis of the matrix P. We took the DFT of each row of P giving the matrix $DFTP$ (Note that this is now a complex-valued matrix). We then took the magnitude of each element of $DFTP$ (resulting in a real-valued matrix), before summing up the rows[8], giving us the real-valued vector $avgDFT$ of length 21. We then plotted $avgDFT$

[7]For more information about Discrete Fourier Transform, any undergraduate digital signal processing textbook should cover the topic in details.

[8]The reason why we took the magnitude of $DFTP$ before summing up the rows, was because we had

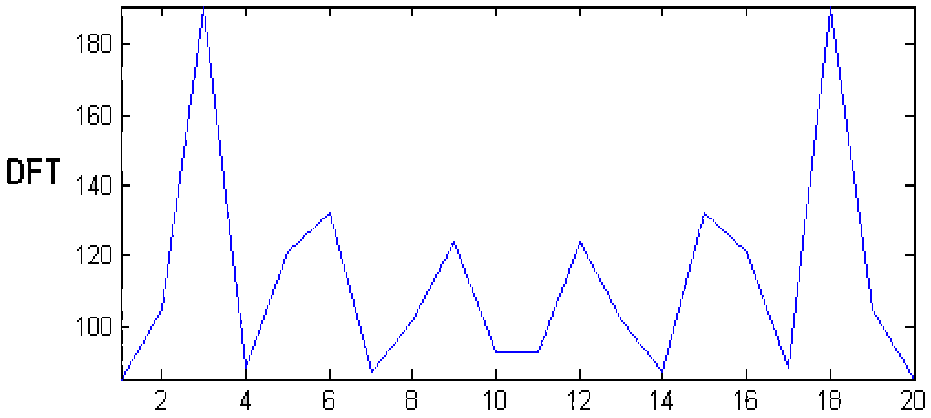

Figure 6: Graph of avgDFT, excluding the first term of the avgDFT.

(see Figure 6), once again excluding the first term of $avgDFT$, and a peak appeared at the position we hoped for.

The two smaller peaks before the half-way point could either be because of noise, or they could be the higher harmonics of the system because we should not be having a perfect sinusoid anyway (we were assuming that we might have some sort of a decaying sinusoid). Consider the DFT of a decaying sinusoid (figure 7, on the next page). Once again, there is a peak at the third point of the DFT (excluding the first term). However, in this case, the peak dies off slowly on both sides. But there are no extra peaks before the half-way point. Thus the decaying sinusoidal model does not adequately explain the two extra peaks in the DFT of the leucine zipper family. That can be grounds for further studies.

4 Conclusion

We set out to improve the inputs to generic neural networks used in secondary structure prediction. Our idea was to input residues correlated to the residue of interest into the TLUs and basic cells of the neural network instead of simply using residues closest to the residue of interest in the primary sequence. We showed that normalized mutual information was effective in picking out correlated residues in the family of leucine zippers. A big proportion of the correlations we found was at least a few standard deviations above the estimated mean of the mutual information of random generated sequences. Such degree of correlation is too significant to be ignored. We also used the discrete fourier transform to

postulated the rows of P were shifted version of a sinusoid with the same period. In the frequency domain, this would correspond to a phase shift of the complex numbers in each row of the DFT. Simply summing up the rows of $DFTP$ might cause the complex numbers to cancel each other out, resulting in a smaller peak at the third point of the DFT which we hope to see.

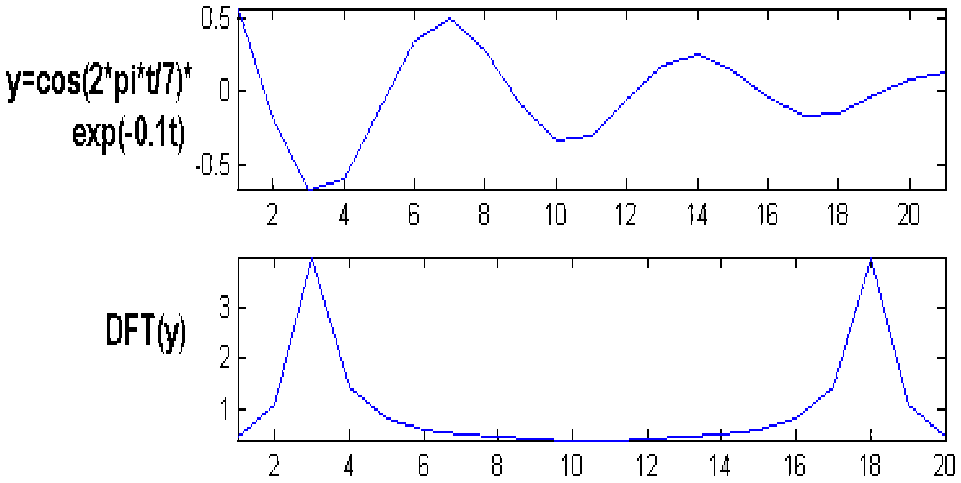

Figure 7: Top: Graph of decaying sinusoid, y. Bottom: Graph of $DFT(y)$, once again excluding the first term.

verify that the normalized mutual information matrix reflected the periodic structure of the leucine zippers. Of course, the ultimate verification of our methods would be to use the PDB database to verify that the normalized mutual information does pick out residues closest to each other in 3-dimensional space. But unfortunately, most of our sequences used in our analysis are not found in the PDB database. Thus, we could not provide any statistics on the accuracy of our methods. The next step in our work would be to actually create a neural network with a pre-processing stage, and actually verify and quantify the improvements we made.

During our investigation of the leucine zippers family, we also discovered interesting higher periodic orders other than the one we expected. Observe that in figure 6 the positions of the smaller peaks (position 6, 9) correspond to exact multiples of the position of the big peak (position 3). An interesting study would be to analyze the significance of these higher harmonics.

5 Acknowledgements

Lastly, I would like to thank Professor Brutlag for the constructive suggestions and guidance he provided through email.

6 References

Blais A, Mertz D. An introduction to neural networks.
http://gnosis.cx/publish/programming/neural_networks.htm

Cover TM, Thomas JA. Elements of Information Theory. A Wiley-Interscience Publication. New York, 1991.

Cuff JA, Barton GJ. Application of multiple sequence alignment profiles to improve protein secondary structure prediction. Proteins 2000 Aug 15; 40(3) : 502 − 11.

Rost B, Sander C. Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 1993 Jul 20; 232(2) : 584 − 99.

Przybylski D, Rost B. Alignments grow, secondary structure prediction improves. Proteins 2002 Feb 1; 46(2) : 197 − 205.

Segars P. Quantitative Analysis of tRNA Sequences Using Information Theory. Biochem 218 Final Project, 2001.

Schneider T. Molecular Information Theory and the Theory of Molecular Machines.
http://www.lecb.ncifcrf.gov/ toms/

7 Appendices

7.1 Appendix 1: Family of Leucine Zippers

Z: refer to a gap in the alignment and appears in leucine zippers of length 22 or 15.

L Q V E N R R L E E Q I K N L T A K K E R L Z Z Z Z Z Z Z
L E E Q I K N L T A K K E R L Q L L N A Q L Z Z Z Z Z Z Z
L E E K V K T L K A Q N S E L A S T A N M L R E Q V A Q L
L E N R V A V L E N Q N K T L I E E L K T L Z Z Z Z Z Z Z
L E K K A E D L S S L N G Q L Q S E V T L L R N E V A Q L
L Q K E S E K L E S V N A E L K A Q I E E L K N E K Q H L
L T G E C K E L E K K N E A L K E R A D S L A K E I Q Y L
L T G E C K E L E K K N E A L K E R A D S L A K E I Q Y L
L T G E C K E L E K K N E A L K E K A D S L A K E I Q Y L
L E R D Y G V L K S N F D A L K R N R D S L Q R D N D S L
L E K D Y G V L K T Q Y D S L R H N F D S L R R D N E S L
L E C E I R K L V C E K E K L L S E R N H L K A C M G E L
L T A E N E R L Q K K V E Q L S R E L S T L R N L F K Q L
L S A E N E K L H Q R V E Q L T R D L A G L R Q F F K K L
Y M A E N E R L R S R V D Q L T Q E L D T L R N L F R Q I
L N Q D R D H L E S E R K R I S N K F A M L Z Z Z Z Z Z Z
L E N R V A V L E N Q N K T L I E E L K A L Z Z Z Z Z Z Z
L E K S A K E M S E K V T Q L E G R I Q A L E T E N K W L
L A Q K V S E L T A A N G T L R S E L D Q L Z Z Z Z Z Z Z
L Q A E T D Q L E D E K S A L Q T E I A N L L K E K E K L
L E D K V E E L L S K N Y H L E N E V A R L Z Z Z Z Z Z Z
L K E Q I R E L V E K N S Q L E R E N T L L Z Z Z Z Z Z Z
L E R K C S L L E N L L N S V Z Z Z Z Z Z Z Z Z Z Z Z Z
L K R C V E K L T E E N R R L E K E A A E L Z Z Z Z Z Z Z
L K R C V E K L T E E N R R L Q K E A M E L Z Z Z Z Z Z Z
L K R C C E N L T D E N R R L Q K E V S E L Z Z Z Z Z Z Z
L R R C C E N L T E E N R R L Q K E V T E L Z Z Z Z Z Z Z
L E R D Y D L L K S T Y D Q L L S N Y D S I V M D N D K L
L E R D Y D S L K K Q F D V L K S D N D S L L A H N K K L
L K K C C E T L A D E N I R L Q K E I Q E L Z Z Z Z Z Z Z
I A I R A S F L E K E N S A L R Q E V A D L Z Z Z Z Z Z Z
L K R C C E S L T E E N R R L Q K E V K E L Z Z Z Z Z Z Z
L K K C C E T L T D E N R R L Q K E L Q D L Z Z Z Z Z Z Z
L L R Y N A N L T K M K N T L I S A S Q Q L Z Z Z Z Z Z Z
L L R Y N A N L T K M K N T L I S A S Q Q L Z Z Z Z Z Z Z

L D A A L H A L Q E E Q A R L K M R L W D L Z Z Z Z Z Z Z
L E N E K T Q L I Q Q V E Q L K Q E V S R L Z Z Z Z Z Z Z
L Q Q E V E K L A R E N S S M R L E L D A L R S K Y E A L
H Q Q D I D D L K R Q N A L L E Q Q V R A L Z Z Z Z Z Z Z
L E S S I H E L T E I A A S L Q K R I H T L E T E N K L L
L N T Q I N K L R D R I E Q L N K E N E F W Z Z Z Z Z Z Z
L D S S I S T L K S R I K E L V N A G C N L S T L N I T L
L I S E E D L L R K R R E Q L K H K L E Q L Z Z Z Z Z Z Z
L E R E L E R L T N E R E R L L R A R G E A D R T L E V M
L E A E R A R L A A Q L D A L R A E V A R L Z Z Z Z Z Z Z
L E A E R A R L A A Q L D A L R A E V A R L Z Z Z Z Z Z Z
L V T C L H L L K L A N K Q L S D K I S C L Z Z Z Z Z Z Z
L S S L S L A V S R S T D E L E I I D E Y I K E N G F G L
L S S L S L T V S R T T D E L E I I D E Y I K E N G F G L
L L N L V E Q L G E A D E E L A G K Y A E L Z Z Z Z Z Z Z
L L C L V E Q L G E E D E E L A G R Y A Q L Z Z Z Z Z Z Z
L L N L V E Q L G E E D E E M T D K Y A E L Z Z Z Z Z Z Z
L Q H S N Q K L K Q E N L S L R T A V H K S Z Z Z Z Z Z Z
L Q Q V N H K L R Q E N M V L K L A N Q K N Z Z Z Z Z Z Z
L E K R V E Y L E T E N T K L V K T L N S L N S E F L Q L
L E K E N T A L R T E V A E L Z Z Z Z Z Z Z Z Z Z Z Z Z Z
L K E Q I R E L A E R N A A L E Q E N G L L Z Z Z Z Z Z Z
L K E H I R D L A E R N A A L E Q E N G L L Z Z Z Z Z Z Z
L E S E K N Q L L Q Q V E H L K Q E I S R L Z Z Z Z Z Z Z
L K E Q I K E L I E K N S Q L E Q E N N L L Z Z Z Z Z Z Z
L K E Q I K E L I E K N S Q L E Q E N N L L Z Z Z Z Z Z Z
L Q Q E R A G L Q E E A Q Q L R D E I E E L Z Z Z Z Z Z Z

7.2 Appendix 2: NormM

Only showing the top upper triangle, and truncating the values to
1 decimal place.

```
2.2 1.3 0.4 0.5 0.4 1.5 -0.3 0.2 0.7 0.1 -0.2 0.3 1.4 -0.4 0.5 0.1 0.4 0.0  0.9 0.8 -0.5
    4.7 6.1 3.5 2.8 2.7  1.6 3.7 5.0 4.2  4.4 3.9 2.1  0.7 4.6 4.5 3.9 3.4  2.3 5.3  1.3
        4.9 4.7 4.3 4.1  0.8 3.5 3.2 2.9  3.0 4.3 3.0  0.2 3.4 4.4 1.8 4.6  3.1 4.8  1.5
            6.0 5.5 4.7  1.6 5.1 3.1 4.2  4.5 6.1 4.7  0.4 4.6 4.9 4.8 5.6  4.0 4.9  1.7
                5.0 4.3  2.3 4.0 3.3 3.5  6.1 3.7 2.7  0.5 4.1 4.4 3.6 4.4  4.1 4.8  1.6
                    3.5  2.7 3.0 3.4 4.3  4.1 3.3 3.0  0.4 3.5 3.0 3.0 3.6  2.3 4.0  3.0
                         0.3 2.4 2.2 3.9  3.9 3.4 3.4  1.1 3.1 3.0 2.7 2.4  2.2 3.6  0.5
                             2.1 1.3 2.2  2.9 1.1 0.5  0.3 0.4 2.6 3.3 2.7  0.7 2.3  4.1
                                 2.4 2.7  4.0 3.6 5.7  0.3 3.3 4.8 3.2 2.0  2.7 4.4  1.2
                                     2.0  1.7 3.1 2.8  0.5 4.8 4.7 3.1 1.4  2.1 4.8  2.0
                                          3.0 3.5 3.0  0.5 2.6 3.2 3.0 2.5  0.9 4.0  2.2
                                              3.4 2.6  1.1 3.7 4.4 4.3 5.5  2.4 3.8  2.2
                                                  3.4  0.3 4.0 3.8 3.7 1.7  3.2 3.6  2.0
                                                       0.6 4.5 4.4 2.3 3.0  3.9 4.2  1.0
                                                           0.4 0.5 0.0 0.4 -0.6 0.1 -0.1
                                                               3.3 2.5 3.6  2.8 3.1  1.7
                                                                   2.9 3.1  2.1 4.0  1.7
                                                                       3.4  2.5 3.4  2.8
                                                                            2.1 4.0  1.9
                                                                                3.5  1.0
                                                                                     2.6
```