Siddharth Bhadra-Lobo, Huy Nguyen
ECS 129 Project Option 4, Winter 2016

Kernel Based Protein Sequence Similarity Metric Analysis

Introduction

In protein sequence alignment, the goal is to compute the level of similarity between two protein sequences based on a pre-generated substitution matrix containing all combinations of amino acid substitutions, scored on the probability of occurring in nature. Traditionally, an alignment is done using methods comparable to the Needleman-Wunsch global alignment algorithm, which employs dynamic programming for pairwise comparison of sequence sites, by scoring alignments based on; match, mismatch, and gap penalty values [1]. Scores do not fulfill the triangle inequality in their measure of translational distance between two sequences, and therefore are not metrics. The dynamic programming method does not always guarantee the most optimal solution, due to the differences in parameterization of the gap penalty, as the cost of opening a gap and expansion of said gap, is associated with arbitrary values. Gaps being nearly ubiquitous for sequence comparisons, leads to variable alignment scores depending on how the gaps are dealt with. The kernel based approach forgoes gaps altogether and is proven to be an actual metric in protein sequence comparison.

Created by Stephen Smale and his group at the City University of Hong Kong, the kernel method was applied on Immunological protein sequences [2]. The method does not generate an alignment, as it does not have gaps considered. Instead, it utilizes pairwise comparisons of each position in the first sequence, to all positions in the second sequence, while referencing the values in the BLOSUM62 substitution matrix. These values are then used to generate the sum of multiplicative pairwise comparisons of k-mer subgroups of increasing size, undergoing the same pairwise comparison method from the previous single position comparisons. The sum is then normalized using the dot product distance of the two sequences, with a final score of zero indicating a perfect match.

This report will present and summarize the kernel method, expected behavior from various test cases, and its current applications with implications of future utility.

Methods

Table1: Top-Down Methods

Input	BLOSUM62.txt	seq1.fa, seq2.fa, seq3.fa	seq1.fa vs seq2.fa	seq1.fa vs seq3.fa	seq3.fa vs seq2.fa
Processing	2-d vectorized	grab sequences	generate K_2 , $\hat{K}_3$, d	generate K_2 , $\hat{K}_3$, d	generate K_2 , $\hat{K}_3$, d
Output	K_1 lookup table	two strings w/o headers	$d(seq1, seq2)$ = 0.278 = **0.2599**	$d(seq1, seq3)$ = 0.0245 = **0.02456**	$d(seq3, seq2)$ = 0.2799 = **0.2614**

Table1: Input files are processed and used to generate K values. Final output of sequence comparisons is shown as distance value, d. The top d-value is from the assignment spec sheet expected final values, the bottom d-value is generated from our program (in bold).

Input: BLOSUM62.txt, seq1.fa, seq2.fa, seq3.fa.

Input consists of 3 files. The first is the BLOSUM62 substitution matrix containing the substitution values of replacing any amino acid for another in sequence. The matrix was stored in a 2-d vector. and it is read into the program line by line. The first column and row contain the single letter codes for the amino acids and their corresponding substitution values in the remaining rows and columns.

Of the three sequence files, only two are read in at a time, as the program only does pairwise sequence comparison. Fasta is the standard format for raw sequence data and appears as a header line beginning with a '>' and followed by information pertaining to the sequence such as, organismal origin, type of sequence, sequencing method used, etc. The line following the header contains the raw sequence data, this is the line read into the program. In larger data sets, the header/sequence lines will repeat for each consecutive read generated during sequencing. The sequence files will be compared two at a time, in a total of three combinations.

Processing: Generation of kernel values (K_1 , K_2 , K_3) and normalizations ($\hat{K}_3$, d).

The position of row zero and column zero of the lookup table have a value overlap at their respective zeroth positions (both are amino acid 'A'). Therefore, columns and rows have to be iterated starting at (i) and (j+1), respectively, in effect ignoring the first repeated value of column zero. For the two sequence files, the header lines (beginning with '>') are ignored, and only the raw sequence is read in as a string. The sequence lengths are recorded and compared, the shorter of the two sequences is then used as the source of positional comparisons. The source sequence will be referred at seq1, the other will be seq2.

K_1 is calculated by doing positional comparisons of the first position in seq1 to every position in seq2, then iterating to the next position in seq1 and repeating comparisons to every position in seq2, until the end of seq1. The corresponding comparison values are done via lookup in the BLOSUM62 substitution matrix. Each comparison value is raised to the power of beta ($\beta = 0.01$), to transform the size of the values generated from the substitution matrix to a smaller value.

$$K_1(x,y) = BL62(x,y)^{\beta}; \qquad (1)$$

where BL62 is the substitution matrix, x and y are positions in seq1 and seq2 respectively. All K_1 values are stored in the 2-d array data structure, with an access time of $O(n^2)$.

K_2 is calculated from the product of K_1 values, in k-mer subgroups u and v, from seq1 and seq2 respectively, ranging in sizes of $i \Rightarrow l$ (meaning i going to l), then the products of subgroup comparisons are summed together per iteration of $i \Rightarrow l$. Subgroups u and v are compared, and equal in size to the current i value, as (u_i, v_i).

$$K_2^l(u,v) = \prod_{i=1}^{l} K_1(u_i, v_i); \quad (2)$$

where $1 \le i \le l$ and $l = 15$ in our implementation. Every consecutive subgroup of u_i is compared with v_i, to generate a K_2 value for each of $i \Rightarrow l$. K_2 values in the program are recursively calculated, as the value for $K_2^{i=3}$ is the product of $K_2^{i=2}$ and the K_1 comparison value of the next position in the sequence, which is accessed from the 2-d array data structure. K_2^i is updated after each recursive step until $K_2^{i=l}$.

K_3 is the sum of all K_2^l values. Meaning K_3 is the sum of all the subgroup positional products of K_1, between groups of size of $i \Rightarrow l$.

$$K_3(s1,s2) = \sum_{u \subset f,\, v \subset g} K_2^l(u,v); \quad (3)$$

where $s1$ and $s2$ are seq1 and seq2, u and v subsequences of $s1$ and $s2$, respectively.

$\hat{K}_3$ is the normalization for calculating the distance between seq1 and seq2, in which the $K_3(s1,s2)$ needs to be become independent to the potential differences in length between seq1 and seq2. Therefore, $K_3(s1,s1)$ and $K_3(s2,s2)$ need to be generated following the same method outlined above, where seq1 and seq2 are compared to themselves. The K_1 values for these comparisons are also stored in their own 2-d array data structure.

$$\widehat{K}_3(s1,s2) = \frac{K_3(s1,s2)}{\sqrt{K_3(s1,s1) \cdot K_3(s2,s2)}}; \quad (4)$$

Dividing $K_3(s1,s2)$ by the square root of $K_3(s1,s1) \cdot K_3(s2,s2)$, produces a value smaller in magnitude with higher sequence similarity, rather than growing larger, to be used to calculate sequence distance. When $s1$ and $s2$ are identical, $\widehat{K}_3(s1,s2)$ evaluates to 1.

Since $\widehat{K}_3(s1,s2)$ is the complement of distance between sequences, per sequence.

$$\widehat{K}_3(s1,s2) = \frac{1 - d^2(s1,s2)}{2}; \quad (5)$$

this can be rearranged to solve for the distance as follows.

$$d = \sqrt{2 - 2\widehat{K}_3(s1,s2)}; \quad (6)$$

where d is the final value for translational distance between $s1$ and $s2$.

Output: Final distance value, d.

From the series of multiplications, and then summations of those products, with input values from the BLOSUM62 substitution matrix being positive decimal values with increasing value the more likely the substitution is to occur in nature, the summations grow with respect to sequence length and similarity. Post-normalization, a smaller d value reflects higher similarity between seq1 and seq2 while a larger value [approaching $\sqrt{2} \approx 1.4142...$ (Table 2d)], reflects sequence dissimilarity, shown in equation (6).

Results

The deviation in d-value from the assignment spec sheet (Table1.) is likely due to our program not truncating source values from the BLOSUM62 substitution matrix, nor any calculated values passed into subsequent calculations. The additional places in the calculations, being incorporated into numerous products and sums, potentially leads to a significant difference in final output values.

To check for expected behavior, two artificial sets of protein sequences, with varying length and degree of polymorphism, were made to create a predictable sequence space, and used as input. Test cases {A,B,C,D,E,F} were made to view how sequence length, interspersed and clustered polymorphisms, affected the final output when compared against sequence A. Polymorphisms with Cysteine (C) and Tryptophan (W) were avoided as their structures lead to amino acids having higher substitution biases, and substitutions were kept consistent across test cases (S->K, N->K). Substitutions of Leucine (L) with Isoleucine (I) and Tyrosine (Y) for Phenylalanine (F), were used for standardization in the number of substitutions between interspersed and clustered substitutions, as it is a relatively low effect substitution and would not skew the final scores as heavily, due to differential polymorphic sites between two comparisons while trying to maintain a consistent number of polymorphism between test cases (e.g. A/E vs A/F).

Table 2a: Short Sequences

Test Case	Amino Acid Sequence	Length	Polymorphic Sites vs. A	d-value
A	SYIRTYLFVENYESL	15	0	0
B	SYIRTYLFVENYESL	15	0	0
C	**K**YIRT**F**L**Y**VEN**F**E**K**L	15	5 interspersed	0.0534
D	**K**YIRT**FI**FVENYE**KI**	15	5 clustered	0.0612
E	**K**Y**L**RT**FIY**VE**KF**E**K**L	15	8 interspersed	0.0781
F	**KFL**RT**F**L**Y**VE**K**YE**KI**	15	8 clustered	0.0793

Table 2a: Sequence A and B are identical. Sequence C includes five polymorphisms (in red/bold) spaced out, and D has five polymorphisms grouped together to represents insertions. E and F follow the same scheme, with nine polymorphisms instead. All sequences are compared against A.

Based on the values generated, it appears there is a lower limit on precision of the sequence similarity, based on length. As sequence A and B are identical, we expect to see zero for the d-value, since equation (4) evaluates to zero with identical sequences. When polymorphisms are clustered, the comparisons are considered more dissimilar than with polymorphisms that are interspersed. This is probably due to the kernel method employing k-mer subgroup comparisons, as a k-mer subgroup will mismatch entire subgroups/significant

proportions of subgroups, during sequence comparison when the polymorphisms are clustered, rather than mismatching fewer positions when they are interspersed. Sequence C and D show a greater d-value difference (0.0078) than with seq E and F (0.0012). This is likely due to the number of polymorphic sites being slightly greater than half of the sequence length and that the polymorphic sites are different, though somewhat accounted for with the specifically chosen substitutions of Leucine for Isoleucine and Tyrosine for Phenylalanine.

Table 2b: Long Sequences

Test Case	Amino Acid Sequence	Length	Polymorphic Sites vs. A	d-value
A long	SYIRTYLFVENYESLSYIRTY LFVENYESLSYIRTYLFVEN YESLSYIRTYLFVENYESLS YIRTYLFVENYESLSYIRTYL FVENYESLSYIRTYLFVENY ESLSYIRTYLFVENYESL	120	0	0
B long	SYIRTYLFVENYESLSYIRTY LFVENYESLSYIRTYLFVEN YESLSYIRTYLFVENYESLS YIRTYLFVENYESLSYIRTYL FVENYESLSYIRTYLFVENY ESLSYIRTYLFVENYESL	120	0	0
C long	KYIRTFLYVENFEKLKYIRTF LYVENFEKLKYIRTFLYVEN FEKLKYIRTFLYVENFEKLK YIRTFLYVENFEKLKYIRTFL YVENFEKLKYIRTFLYVENF EKLKYIRTFLYVENFEKL	120	40 interspersed	0.2086
D long	KYIRTFIFVENYEKIKYIRTFI FVENYEKIKYIRTFIFVENYE KIKYIRTFIFVENYEKIKYIRT FIFVENYEKIKYIRTFIFVENY EKIKYIRTFIFVENYEKIKYIR TFIFVENYEKI	120	40 clustered	0.2140
E long	KYLRTFIYVEKFEKLKYLRT FIYVEKFEKLKYLRTFIYVEK FEKLKYLRTFIYVEKFEKLK YLRTFIYVEKFEKLKYLRTFI YVEKFEKLKYLRTFIYVEKF EKLKYLRTFIYVEKFEKL	120	64 interspersed	0.2567
F long	KFLRTFLYVEKYEKIKFLRT	120	64 clustered	0.2519

	FLYVEKYEKIKFLRTFLYVE KYEKIKFLRTFLYVEKYEKI KFLRTFLYVEKYEKIKFLRT FLYVEKYEKIKFLRTFLYVE KYEKIKFLRTFLYVEKYEKI			

Table 2b: Same as Table2a, above, but with sequences lengths eight times longer, even longer than seq1-3 sequences used as input.

With the longer test sequences, the same trend with the short sequences is observed, that the interspersed polymorphisms show greater sequence similarity than the clustered polymorphisms, with sequences C long and D long. Even though the proportion of polymorphism is the same across short sequences and long sequences (1/3 interspersed and 8/15 clustered), the d-value difference for short E and short F is less significant (~0.001, Table 2a) than for long E and long F (~ -0.005, Table 2b), where long F is actually measured to be more similar to long A than long E is to A. We expect the F d-values to be larger than E, in both short and long sequences. This is potentially due to the number of subgroups used for a short sequence versus a long sequence, since there is only one subgroup of size fifteen for the short sequences, while there are many of that size for the long sequences. Therefore the weight associated with the smaller subgroups is higher in short sequences, leading to lower perceived sequence similarity when polymorphisms are clustered in groups for short sequences. The opposite behavior is observed when a greater proportion of polymorphisms (8/15) are present in a sequence, or when the sequence length is much longer than the maximum subgroup size used. This will be further tested in Table 2c. The kernel method does subgroup-subgroup comparisons, and when an entire subgroup or significant portion of the subgroup mismatches between the two sequences. This can lead to larger differences in calculated d-value, since the K_3 depends on the sum of the single position comparison products in subgroups and small deviations in value can propagate into the final value as many of these comparisons are made, as shown in equations (2) and (3).

Table 2c: Confirmation Sequences

Test Case	Amino Acid Sequence	Length	Polymorphic Sites vs. respective G	d-value
G	IIIIIIIIIIIIIII	15	0	0
H	IIIIIIIIIIIIIII	15	0	0
I	LIIIILILIIILILI	15	5 interspersed	0.08937
J	LIIIILLIIIIILL	15	5 clustered	0.07811
K	LILIILLLIILLILI	15	8 interspersed	0.15087

L	LLLIILILIILIILL	15	8 clustered	0.12496
G long	IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	120	0	0
H long	IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	120	0	0
I long	LIIIILILIIILILILIIIILILIIILILILIIIILILIIILILILIIIILILIIILILILIIIILILIIILILILIIIILILIIILILILIIIILILIIILILILIIIILILIIILILI	120	40 interspersed	0.27011
J long	LIIIILLIIIIIILLLIIIILLIIIIIILLLIIIILLIIIIIILLLIIIILLIIIIIILLLIIIILLIIIIIILLLLIIIILLIIIIIILLLIIIILLIIIIIILLLIIIILLIIIIIILL	120	40 clustered	0.27021
K long	LILIILLLIILLILILILIILLLIILLILILILIILLLIILLILILILIILLLIILLILILILIILLLIILLILILILIILLLIILLILILILIILLLIILLILILILIILLIILLILILILIILLLIILLILI	120	64 interspersed	0.41527
L long	LLLIILILIILIILLLLLIILILIILIILLLLLIILILIILIILLLLLIILILIILIILLLLLIILILIILIILLLLLIILILIILIILLLLLIILILIILIILLLLLIILILIILIILL	120	64 clustered	0.41382

Table 2c: Both short and long sequences, length fifteen and 120 respectively. Where polymorphic sites are positionally equivalent to Table 2a and Table 2b.

Since the previous test cases produced unexpected d-values between interspersed and clustered polymorphic sites, especially for the 8/15 polymorphic proportion sequences, particularly with the long sequences, test cases {G,H,I,J,K,L} were created. These sequences only use two amino acids, Isoleucine (I) and Leucine (L) to minimize the d-value differences that could accumulate from the positional effects of having different amino acids being polymorphic across test sequences. Surprisingly, across both lengths, the clustered polymorphisms are measured as being more similar to the respective sequence G. With short I and J the difference (0.01126) between clustered is not as large as short K and L (0.02591), but still consistent with test cases {A,B,C,D,E,F}, that clustered polymorphisms actually produce higher perceived sequence similarity. The differences in d-value for the long sequences are relatively small (~0.0001 for I and J, 0.0015 for K and L) but still show that the clustered polymorphisms are perceived slightly more similar to sequence G long. This is potentially due to the way subgroups are used in comparisons, as discussed in the previous section, such that

when the subgroup comparisons are made, the clusters affect the scores of subgroups containing the polymorphisms, but also leaves larger clusters of non-polymorphic regions. The non-polymorphic regions maintain integrity as a result of the clustered polymorphism. Therefore more subgroup comparisons are made with whole regions being non-polymorphic, or significant portion being non-polymorphic, as opposed to the interspersed regions having more single polymorphisms spread out, thereby affecting the score of multiple individual subgroup comparisons.

Table 2d: Dissimilar Sequences

Test Case	Amino Acid Sequence	Length	Polymorphic Sites vs. respective length X	d-value
X	DDDDDDDDDDDDDDD	15	0	0
Y	WWWWWWWWWWWWWWW	15	15	0.5894
X long	DDDDDDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDDDDDD	120	0	0
Y long	WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWWWW	120	120	1.2534
X extra long	Not included for space. 240 Ds	240	0	0
Y extra long	Not included for space. 240 Ws	240	240	1.39466

Table 2d: Sequences X and Y, are of length fifteen and different at every position. Sequence X long and Y long with length 120 and different at every position, and X extra long and Y extra long, with length 240 and different at every position, are used to attempt to calculate the upper limit on the d-value.

(Table 2d) exhibits a test case with the most dissimilar sequences of length fifteen and 120 possible, based on the BLOSUM62 substitution matrix, between Aspartic Acid (D) and Tryptophan(W) $(K_1 = 0.2321)$. The d-value result of the short sequences of 0.5894, is not very close to the upper limit of $\sqrt{2} \approx 1.4142$. With the long sequences, the d-value result of 1.2534

and 1.39466 are approaching the upper limit of $\sqrt{2}$. This is the expected behavior, supported by equations (4) and (6).

Conclusion

The kernel method from Smale et al. appears very promising as it fulfills the criteria for metricality in pairwise protein sequence comparison, is relatively simple to implement, and provides a foundational approach for future expansion in applications.

A current use of this method is done by Gold et al. using the the kernel method for protein sequence comparison, without any alterations to the published method [3]. Fan et al. set the kernel method as a benchmark score for their new clustering tree model, using allele coverage as an advantage, in *ab initio* protein structure prediction [4]. This shows how already established this method is, as it was chosen out of a multitude of potential options, to be used as a trusted source for sequence validation. Fan et al.'s group also included the kernel method in a PhD thesis from their lab, for protein binding affinity prediction, employing potential improvements by running Regularized Least Squares (RLS) learning by incorporating $\hat{K}_3$ values into regression equations.[5] Another use, from Feng et al., was the incorporation of the $\hat{K}_3$ value into a Gaussian kernel for clustering classification of alleles, in which their model uncovered additional outliers in classification that Smale et al. "missed" with their method [6]. This can be considered an improvement on the original kernel method, with the inclusion of the Gaussian kernel, circumventing the complement based d-value for a new D_L.

The modifications of the kernel method alone are a nod towards future applications of the method. By taking $\hat{K}_3$ values as parameters in other mathematical functions, leaves many possibilities for further application, such more accurate homology modeling or *ab initio* protein structure prediction.

One area for improvement of the method was encountered during testing with interspersed vs clustered polymorphisms, where the number and proportions of polymorphisms being conserved between test cases still lead to deviations in the distance values. Biologically, the difference between a clustered polymorphism, such as the addition of foreign DNA loci into the coding region of a gene, could be seen as more significant than point mutations of the same quantity, when only dealing with insertions and not deletions. But in the test cases of artificial the protein sequence, the "neutral" substitution amino acid positions should have accounted for differences in substitution values when clustered or interspersed. The difference is small for the long sequences, potentially negligible, unless doing multiple sequence homology modeling where small domains could be crucial to overall conformation and function. But the short sequences, showed significant distance deviation for relatively similar sequence composition. Therefore, an additional step could be included in generating the K_2 values. While still doing subgroup comparisons, if the K_1 of a certain position is significantly lower than the surrounding K_1 comparisons in its respective subgroup, it can be considered an isolated polymorphism. But if the K_1 is also similarly low in the flanking position K_1 values, then initiate a comparison with the flanking subgroups total K_1 average, with the current subgroups K_1 average. If the K_1 average is below the two flanking subgroups, then the current subgroup is likely a clustered

polymorphism and can be flagged to be associated with a higher substitution weight, and therefore less likely to skew the final sequence comparison score. This can be applied to a variety of subgroup sizes, particularly for smaller subgroups. Just an idea for further thought and testing.

References

[1] S. Needleman, C. Wunsch. **"A general method applicable to the search for similarities in the amino acid sequence of two proteins"**. Journal of Molecular Biology 48 (3): 443–53. (1970).

[2] W.-J. Shen, H.-S. Wong, Q.-W. Xiao, X. Guo, and S. Smale. "**Towards a mathematical foundation of immunology and amino acid chains"**. (2012).

[3] Gold, Marielle C.; McLaren, James E.; Reisetter, Joseph A.; Smyk-Pearson, Sue; Ladell, Kristin; Swarbrick, Gwendolyn M.; Yu, Yik Y L; Hansen, Ted H.; Lund, Ole; Nielsen, Morten; Gerritsen, Bram; Kesmir, Can; Miles, John J.; Lewinsohn, Deborah A.; Price, David A.; and Lewinsohn, David M. **,"MR1-restricted MAIT cells display ligand discrimination and pathogen selectivity through distinct T cell receptor usage",** The Journal of Experimental Medicine.211,8. 1601-1610. (2014).

[4] Y. Fan, R. Lu, L. Wang, M. Andreatta and S. C. Li, **"Quantifying Significance of MHC II Residues".** in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 1, pp. 17-25, Jan.-Feb. (2014).

[5] Andreatta, M., Nielsen, M., & Lund, O. **"Discovering sequence motifs in quantitative and qualitative peptide data"**, Kgs. Lyngby: Technical University of Denmark. 2012.

[6] Feng Tan and J. J. Slotine, **"A quorum sensing inspired algorithm for dynamic clustering".** *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Firenze, pp. 5364-5370. (2013).