Training and Testing of an Hidden Markov Model for finding the presence of the Kunitz domain

Matteo Manfredi
Master in Bioinformatics, University of Bologna

7 may 2018

Abstract

In this paper we show the procedure to build an HMM to predict the presence in new proteins of a domain of interest, the Kunitz/BPTI, starting from a structural alignment between a training set of proteins that contains it.
The generated model was then tested against all known proteins and we demonstrated that it was able to obtain very good levels of accuracy.
In fact, almost every analysed instance was correctly classified, and the only outliers were a few sequences with some particularity that seems to justify the apparent mistake made by the model.

1 Introduction

The goal of this paper is to generate an Hidden Markov Model (HMM) that is able to correctly predict if a newly examined sequence contains or not the Kunitz domain.

Usually, HMM models are generated starting from a sequence alignment between a training set of known proteins.
Since we know that structural information is always more accurate than information at the level of the sequence, especially when dealing with the function of the proteins, for this project we decided to start from a structural alignment to generate our model.

Kunitz domains are the active domains of proteins that inhibit the function of protein degrading enzymes or, more specifically, domains of Kunitz-type are protease inhibitors. They are relatively small with a length of about 50 to 60 amino acids.
Kunitz domains are stable as standalone peptides, able to recognise specific protein structures, and also work as competitive protease inhibitors in their free form. These properties have led to attempts at developing biopharmaceutical drugs from Kunitz domains [1].

2 Material and Methods

2.1 Training Set Selection

For the first step of this project we had to construct a training set of proteins that will be used later to generate our model.
We used the Protein Data Bank (PDB) [2] to search for candidates and we decided to include in the set only proteins that meets the following requisites:

- They contain the Kunitz domain (Pfam [3] Accession Number: PF00014)
- Their sequence length is between 50 and 99 residues (if they were shorter they could

not possibly include the domain, if they were longer they could contain multiple domains)

- Their structure was resolved at a resolution level not higher than 2.5 Å
- They can not contain any modified polymeric residues
- Finally, we were not interested in using sequences too similar, so we wanted to cluster them at 90% level of sequence identity, retrieving only one representative structure for each cluster

To do so, we performed the following query through the advanced search panel :

Pfam Accession Number PF00014 and Sequence Length is between 50 and 99 and Resolution is 2.499 or less and Ligand Search : Has modified polymeric residues=no and Representative Structures at 90% Sequence Identity

Using a small script we then extracted the ID of the selected chains, obtaining a list of 18 items.

2.2 Structural Alignment and Model

After the selection of the training set, we had to perform a structural alignment between all of them. For doing this we used the on-line version of PDBeFold [4] .
From their website, we first of all selected the option *multiple structural alignment*, then we selected *list of pdb codes* for the format of the input (that in our case is the list mentioned before).

Once the computation was complete, we downloaded 3 files from the result page, that will all be attached in the corresponding appendices:

- *match.dat* that contains every statistical information that was extracted by the program, allowing us to perform a first manual check of the quality of the alignment.
- *send.rasmol* that contains the structural informations from every chain used for the alignment, allowing us to visualize it.
- *fasta.seq* that contains the actual alignment.

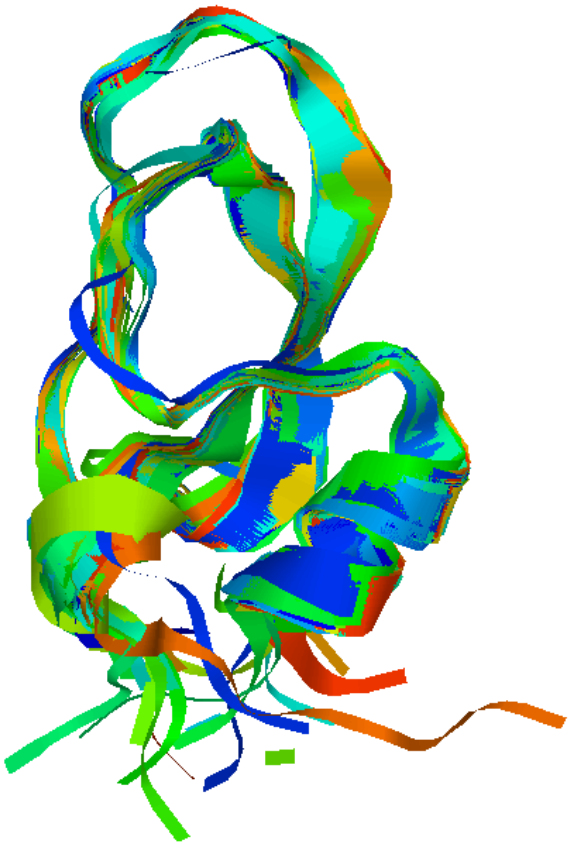

Figure 1: Alignment of all the chains used displayed using RasMol [5] , we can see that they are all very similar.

Using the last mentioned file as input, we were then able to build the desired HMM model with the following command from HMMER

[6] :

hmmbuild model.hmm fasta.seq

At this point we already generated what we needed for this project, but we still had to understand how well it would perform in recognizing the presence of the Kunitz domain given an unknown sequence of residues.

2.3 Benchmark Set

In order to test our model, we had to generate a benchmark set composed of proteins for which we know in advance if they contain or not the Kunitz domain.
We already had downloaded in our machine every sequence from UniProtKB/Swiss-Prot [7] in fasta format, so what we had to do was to extract a positive set and a negative set from here.

From the advanced search panel of UniProt we searched for every reviewed protein containing the Kunitz domain and we downloaded them both as a list of ID and as a multiple *fasta* file, the first needed for obtaining the negative set and the latter being the starting point for building the positive set.

We then wrote a python program that simply takes in input a multiple *fasta* file and a list of ID and generate a copy of the first file, removing if present the fasta sequences corresponding to ID contained in the second file.
Running this program was sufficient to generate the final Negative Set (composed of 556653 sequences), using as input the fasta of all sequences from Swiss-Prot and the downloaded list of ID, but in order to obtain the Positive Set we had more work to do, since we wanted to remove from there sequences too similar (we used a 90.0% threshold of similarity) to the ones used in the Training Set.

To do so we used *blastall* [8] with the option *-p blastp* and we searched all the downloaded fasta against our training set. We then wrote a simple script to extract from the output of blastall a list of 36 ID that had to be removed, so that we could reuse the above mentioned python program to generate the final Positive Set composed of 323 sequences.

At this point, all we had to do was to write another script to mark the positive and the negative sequences and to merge them together in the final benchmark set composed of 556976 sequences.

2.4 Testing

With the final benchmark set that we generated we could finally test our model.

First of all we used the command **hmmsearch** with the options *–noali* (less verbose output), *–max* (no filters used) *-E 1000* (do not display sequences that obtain an e-value higher than 1000) and *-o f* (redirect output to a file f) to generate a file containing information on how probable it is for sequences in the benchmark set to contain the Kunitz domain, based on the used model.

What we did was then to set different e-value thresholds (from 10^{-4} to 10^{+2}) and to count the number of positives and negatives obtained, generating 7 different confusion matrices.
Finally, we wrote a python program to compute performance indexes on those matrices.

3 Results

At the end of the testing phase, we proved that our model is really good at predicting the presence of the Kunitz domain.

The best e-value threshold for the data we used is 10^{-3}, with only 2 false negatives and no false positives.
However, the thresholds used for the testing return all good performance values, even if they differ in many orders of magnitude, meaning that the model is able to make a clear distinction for almost all known sequences, with a very few outliers compared to the huge number of sequences that we tested.

e-value	(TP/FP|FN/TN)		Acc | MCC
10^{-4}	319	0	0.999993
	4	556653	0.993785
10^{-3}	321	0	0.999996
	2	556653	0.996897
10^{-2}	321	1	0.999995
	2	556652	0.995347
10^{-1}	321	4	0.999989
	2	556649	0.990740
1	321	6	0.999986
	2	556647	0.987704
10	323	12	0.999978
	0	556641	0.981916
10^2	323	55	0.999901
	0	556598	0.924345

Table 1: Results of the test for the different e-value used.

3.1 Manual Check

We further inspected our data to check the e-value obtained for the outliers and these were the results:

- The positive instance with the highest e-value is the UniProt entry D3GGZ8, with e-value = 8.6. The second is the UniProt entry O62247 with e-value = 1.5, while the others have all e-value lower than 10^{-3}
- The negative instance with the lowest e-value is the UniProt entry P84555, with e-value = 0.0015, while the others have all e-value higher than 10^{-2}
- There are a total of 12 negative instances with e-value lower than 8.6, meaning that it is impossible to find a threshold that perfectly separates the positives from the negatives

3.2 Outliers

The last thing we did was to check the three above mentioned entries in the UniProt website.

The negative instance P84555 seems to actually have the Kunitz domain, but this was not annotated in the entry with the usual Pfam annotation.

The two positive instances D3GGZ8 and O62247 are instead both marked with the sentence *"Appears to lack serine protease inhibitor activity in vitro when tested with bovine pancreatic alpha-chymotrypsin and elastase"*, which could probably have some correlation with the fact that they are different from the others.

Those findings only confirm the quality of the produced model, that correctly marked those sequences as outliers.

3.3 Appendices

Every attached file can be found in a GitHub repository accessible from the following link:

https://github.com/847776/Lab1Project

Here is displayed the full list, with a brief description of their content:

- *tabularResults.csv* contains the results of the initial query performed on PDB.

- *cdv2idlist.sh* is the script that automatize the procedure of extracting the ID list of the training set. It also prints in output the steps needed to proceed with the creation of the model.
- *training_set.txt* is the output of *cdv2idlist.sh*.
- *fasta.seq, match.dat and send.rasmol* contain the results of PDBeFold.
- *all_negative_set.txt* contain the ID list of all UniProt proteins that contains the kunitz domain. Note that for running the second script we would also need their fasta sequences in a different file and a second fasta file containing all the proteins from Swiss-Prot.
- *remove_fasta.py* is the Python program used by the second script.
- *workflow.sh* is the script that automatize the entire procedure of building the model and the benchmark set, giving in output the file needed for the final testing phase.
- *output.txt* is the output of *workflow.sh*.
- *evaluate_confusion_matrix.py* is the Python program used by the final script.
- *test.sh* is the script that automatize the final testing, printing on the terminal the confusion matrices, the accuracy and the Matthew Correlation Coefficient obtained for different e-value thresholds.

References

[1] *http://pfam.xfam.org/family/PF00014*

[2] *H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne. (2000) The Protein Data Bank Nucleic Acids Research, 28: 235-242*

[3] *Robert D. Finn, Penelope Coggill, Ruth Y. Eberhardt, Sean R. Eddy, Jaina Mistry, Alex L. Mitchell, Simon C. Potter, Marco Punta, Matloob Qureshi, Amaia Sangrador-Vegas, Gustavo A. Salazar, John Tate, Alex Bateman; The Pfam protein families database: towards a more sustainable future, Nucleic Acids Research, Volume 44, Issue D1, 4 January 2016, Pages D279–D285, https://doi.org/10.1093/nar/gkv1344*

[4] *Krissinel E., Henrick K. (2005) Multiple Alignment of Protein Structures in Three Dimensions. In: R. Berthold M., Glen R.C., Diederichs K., Kohlbacher O., Fischer I. (eds) Computational Life Sciences. CompLife 2005. Lecture Notes in Computer Science, vol 3695. Springer, Berlin, Heidelberg*

[5] *Roger Sayle and E. James Milner-White. "RasMol: Biomolecular graphics for all", Trends in Biochemical Sciences (TIBS), September 1995, Vol. 20, No. 9, p. 374*

[6] *http://hmmer.org/*

[7] *The UniProt Consortium; UniProt: the universal protein knowledgebase, Nucleic Acids Research, Volume 45, Issue D1, 4 January 2017, Pages D158–D169, https://doi.org/10.1093/nar/gkw1099*

[8] *http://nebc.nox.ac.uk/bioinformatics/docs/blastall.html*