Alex Han
STAT 229
June 06 2016

Modeling HLA ligands for binding prediction of new peptides

Introduction

The HLA (human leukocyte antigen) is a family of receptor proteins found on the outside of nearly every cell in the human body. They are one of the most important components of the innate immune system. They bind randomly fragmented peptides from discarded proteins floating around the interior of the cell, then pull them out to the cell surface to present them to the immune system. Generally, HLA will bind peptides representing the repertoire of human proteins. But if a virus has infected a cell, HLA will present peptides from viral proteins to the surface of the cell for other cells to detect and trigger an immune response. In this way, the HLA serve as a "window" for the inner contents of a cell to be seen without other cells having to invasively sample it from within. A cartoon depicting the function of is shown in Figure 1.

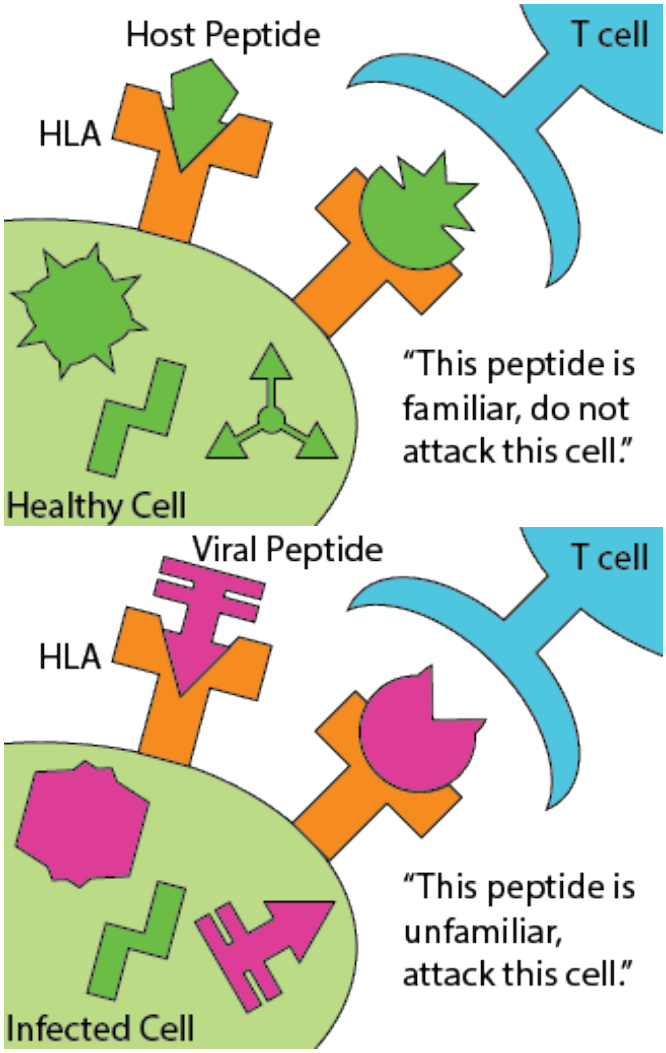

Figure 1: *Simple illustrations of HLA function in both healthy and infected cells.*

Over the course of human history, HLA have evolved to be able to recognize, bind, and present the wide variety of proteins of pathogens, whereas pathogens have evolved to escape detection. Because of this human-pathogen coevolution, HLA is now one of the most hypervariable proteins in humans, with easily over thousands of different known alleles. As a result, each HLA has unique binding properties that allow it to present different peptides, whether host or foreign.

Developments in mass spectrometry have allowed biologists to sequence these peptide ligands that can bind to a single HLA allele in a healthy human cell line. Older assays could only offer a dissociation constant, or a metric of how well a ligand is thought to an HLA without offering direct proof of whether a ligand was actually bound by HLA. A newer method has been developed that definitively shows binding with an HLA allele. However, it does not come with dissociation constants, and we cannot sequence the ligands that did not bind.

Earlier machine learning methods for peptides based on these older experiments relied on a threshold for the dissociation constant and made a two-class problem by labeling peptide ligands as "good binders" and "poor binders/non-binders". However, this new dataset has only has "good binders". Unfortunately, these datasets cannot complement each other as they relied on different experimental protocols.

My two goals in this project were to model the distribution of peptides that bind to each HLA and compare them, and use this data to predict whether untested peptides will bind or not.

Data and Preprocessing

My unpublished experimental dataset has been given to me by Curtis McMurtrey and William Hildebrand. There are peptide sequences from 5 different HLA alleles. In this experiment, I will be using just the peptides that are 9 amino acids long as it has been shown that these "nonamers" are the likely canonical peptides that HLA are best designed to fit. It is important to note that all of these sequences were shown to bind; there is no experimental data for peptides that do not bind. A brief summary of counts of sequences is shown in Table 1.

Table 1: *A short summary of the sequences used.*

HLA	# of nonamers
B*07:01	1488
B*15:01	2836
B*46:01	1339
B*73:01	119
C*01:02	436

For model testing, I randomly generated sequences as

a negative set using the frequency of amino acids found in the human proteome. However, the experimental data has a low sensitivity rate, such that any randomly generated sequence may actually be a binding sequence. However, given the high dimensionality of the data (9^{20} possible nonamers), I am using the assumption that these randomly generate sequences will suffice as nonbinders.

For testing predictions, I will be using a set of sequences also obtained using the same experimental protocol, but with only HLA-B*07:01 in a cell with a strain of Influenza A virus H1N1 (swine flu).[1] A small number of these sequences have been identified as binders from the viral proteome. The remaining set of all possible nonamers from the viral proteome are used as nonbinders.

Rather than use sparse encodings of these peptide sequences (i.e. create 20 binary variables for every 9 positions of the amino acid for the presence of one of the 20 amino acids at every position), I opted to convert the sequences into physiochemical properties, creating a numerical pseudo-continuous range of values. The four properties I used were molecular weight, surface area, isoelectric point, and hydrophobicity index, thus converting every nonamer string into 36 features. Figure 2 explains this transformation. A two-dimensional t-SNE on the transformed experimental data is shown in Figure 3.

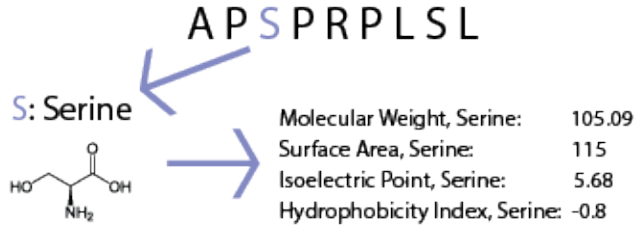

Figure 2: *The 36 features generated using the 9 positions and 4 physiochemical amino acid properties.*

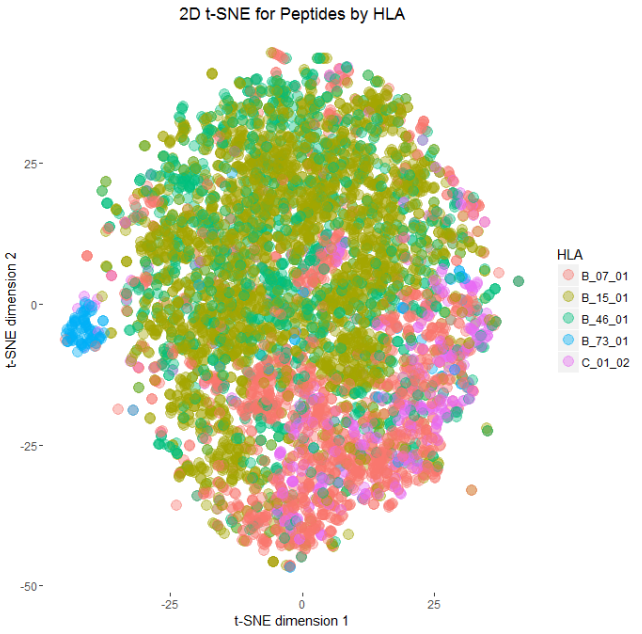

Figure 3: *t-SNE of the experimental data.*

Kernel Density Estimation

Because the data is a one-class set, I sought a generative approach by building a model that would best explain the sequences. I settled on using multivariate kernel density estimation for this purpose. It addresses several quirks with the data including its irregular multivariate distribution, sparseness in its dimensions, and its high covariation among not only just the 4 physiochemical properties assigned to each position, but also across positions. I could then later fit new peptides into this model the likelihood of binding with an HLA.

However, the significant drawback of this approach is its complexity. Because creating a density in 36 dimensions is prohibitively costly, I decided to first reduce the dimensionality of the data. I tried two approaches: principal components analysis and autoencoders.

PCA + KDE

PCA is a simple linear transformation of the original variables that also maximizes the total variance in its first few components. So, the first few principal components can be used as a substitute for the original dataset. This method's downsides are that PCA cannot capture nonlinear trends, and it misses out on the lesser components, which while less important, represent unique aspects of the data.

After PCA, I chose the first five 5 principal components of the data use as my features for the multivariate KDE, which accounted for approximately 40-60% of the total variance depending on the HLA allele.

To verify the results of this method, I used 10-fold cross validation for the dataset, in addition to using 1000 randomly generated peptide sequences. They were then plugged back into the KDE after conversion with the original loading matrix to obtain likelihood values. Any conclusions drawn from their model fit should be taken cautiously as some of these random sequences may very possible bind to the HLA in reality.

At each iteration of the cross validation, I used 90% of the sequences' PCs to construct the KDE model, then the reamining 10% and the fake peptides to test the model. The bandwidth matrix of the KDE was optimized using the smoothed asymptotic mean squared error (SAMSE).[2]

Autoencoder + KDE

Autoencoders are neural networks that aims to create an output approximate to its input after passing through a hidden layer with fewer nodes. The hidden layer encoding with fewer nodes can then be used to substitute the original dataset. This method's downsides are that it relies on neural networks, which not only have many input parameters, but also do not underly a probabilisitic foundation, which means optimizing and evaluating a neural network, is difficult to do empirically.

Instead of using a PCA and choosing the top 5 principal components to use for the KDE, an autoencoder with one hidden layer with 5 nodes was first constructed using the training set of 90% of the true peptides. The autoen-

coders were manually tried under a variety of parameters to assure optimal information retention and feature diversity among hidden nodes. Similar to the PCA + KDE approach, 90% of the sequences' PCs to construct the KDE model, then the reamining 10% and the fake peptides to test the model.

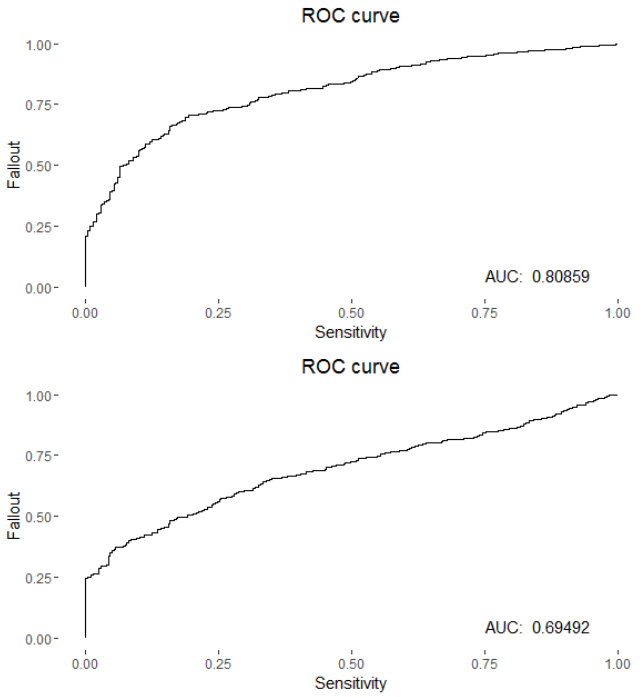

Figure 4: *Example ROC curves, for the same samples shown in Figure 2 (top) and Figure 3 (bottom).*

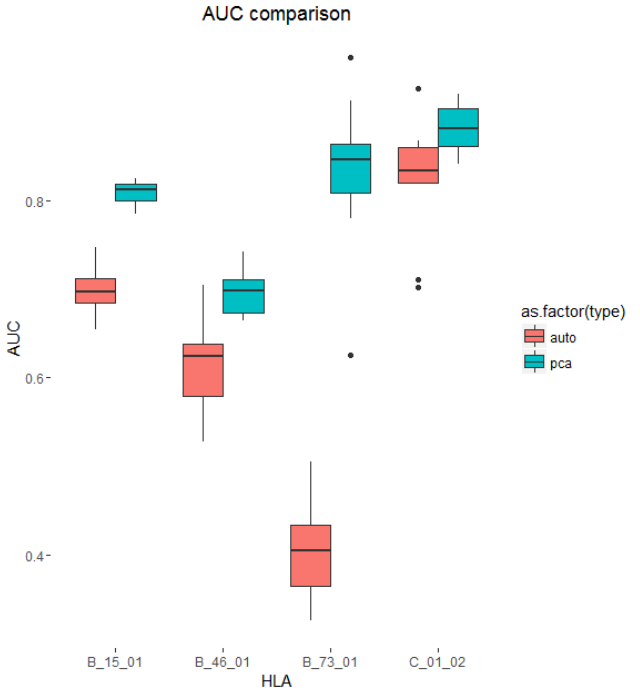

Figure 5: *Box plots of AUC values for 10-fold cross validation results.*

KDE performance

To visualize the overall predictive performance, I created ROC curves to visualize and compare the accuracy of these methods. The ROC curves in addition to their AUC values for a single k-folds iteration is shown in Figure 4.

For each of the k-folds iterations for both methods, an AUC score was calculated. These AUC scores were then averaged over the ten iterations. These values are plotted in Figure 5.

Sampled Kullback-Leibler divergences

The two sets of KDE likelihood distributions were also used to study the difference among HLA probability distributions. Rather than use the empirical distribution of 20^9 possible sequences, I used randomly generated sequences to obtain sampled Kullback-Leibler divergences. The matrix of KL divergences is shown in Table 2 for the PCA KDE and Table 3 for the autoencoder KDE.

Table 2: *A matrix of Kullback Leibler divergences for the PCA + KDE approach*

HLA	B*07:01	B*15:01	B*46:01	B*73:01	C*01:02
B*07:01	0	0.51	0.24	0.40	0.83
B*15:01	0.37	0	0.05	0.24	0.53
B*46:01	0.47	0.18	0	0.29	0.58
B*73:01	0.19	0.18	0.09	0	0.30
C*01:02	0.25	0.15	0.07	0.17	0

Table 3: *A matrix of Kullback Leibler divergences for the Auto + KDE approach*

HLA	B*07:01	B*15:01	B*46:01	B*73:01	C*01:02
B*07:01	0	13.8	12.3	10.8	12.2
B*15:01	9.69	0	2.62	4.68	1.80
B*46:01	29.7	14.2	0	13.9	10.3
B*73:01	540	401	402	0	335
C*01:02	80.9	39.9	31.5	22.9	0

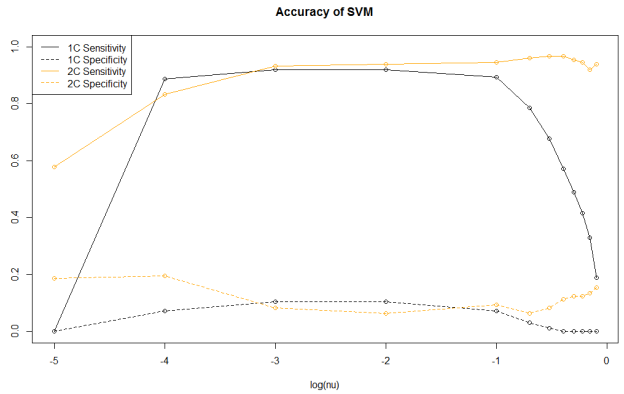

Figure 6: *Accuracy of SVM methods by varying ν.*

One- and two-class ν-support vector machines

A simpler prediction model was created using SVM. The one-class SVM was trained on 90% of the experimental sequences, while the two-class SVM used both the

90% experimental sequences and 90% of the generated sequences. They were both tested with the unused 10% of both sets of sequences. Rather than test the overall accuracy of the results, I tested the sensitivity and specificity rates of the test set. The results are plotted across various values of ν in Figure 6.

Prediction of viral proteome binding

To test the accuracy of all four predictions methods on an actual dataset, I first chose the optimal likelihood threshold using all available data, and chose optimal values of ν for the SVM models with all the data, to attain the best overall accuracy. Then, all possible 9mers from the swine flu virus strain were tested. The sensitivity and accuracy of these predictions are shown in Table 4.

Table 4: *Prediction of H1N1 proteome binding*

	PCA KDE	Auto KDE	1-SVM	2-SVM
Sensitivity	1	1	1	1
Specificity	0.915	0.531	0.100	0.154

Discussion

With regards to comparing the distributions of HLA, t-SNE appears to "perform" the best, even though it is not an explicit clustering method. PCA and autoencoding of the data do not show distinct clusters as well as t-SNE, and an approach with k-means in varying numbers of dimensions was unable to return any meaningful classifications. This is likely because the data is sparse and has correlations in higher dimensions. Sampled KL distances from the KDE distributions does not necessarily agree with the t-SNE interpretation of clusters, though it is a very different metric to compare with. From t-SNE, it would appear that there are three meaningful clusters of the five HLA: {B*07:01, C*01:02}, {B*15:01, B*46:01}, and {B*73:01}.

In my KDE approaches, PCA approach tends to fare better than the autoencoder approach across the board. This may largely be because of the small sample size of this dataset. PCA tends to be more robust in this situation.

The SVM models to predictably poorly as with the experimental data and random sequences in the sine flu virus peptide prediction. This may because of the high dimensionality of the dataset and small sample size as well.

PCA+KDE in general appears to perform the best in terms of prediction capability. In fact, it is very close to the accuracy of predictions run through one of the leading peptide prediction algorithms, netMHC.[3,4] However, netMHC relies on a much larger database of peptide sequences of an older method, so it is not a completely fair comparison.

Future directions

I will try to use the set of sequences from the larger databses of peptides to see if binding predictions can be improved. These data however, as mentioned in the introduction, use a different method that cannot explicitly tell whether a peptide had bound or not. At the very least, recreating models with the larger database can shed light on whether the newer deep ligand sequencing is on par with the older information.

I am also looking into the accuracy of other simpler baseline methods by using a combination of dimension reduction and feature selection with SVM or random forests.

Acknowledgements

I would like to thank Curtis McMurtrey and William Hildebrand for the sequencing data and information regarding the technicalities of deep ligand sequencing.

I also thank the Parham and Bustamante labs, in particular, Hugo Hilton for regular discussions on this project, and Elena Sorokin for suggestions on machine learning approaches.

Lastly, I'd like to thank John Duchi and the TAs of STAT 229, in particular, Claire, for teaching this incredibly challenging and rewarding course.

References

[1] Wahl A. et al. (2009) *HLA class I molecules consistently present internal influenza epitopes* PNAS. 106 (2), 540-545

[2] Duong T. and Hazelton M.L. (2003) *Gapped sequence alignment using artificial neural networks: application to the MHC class I system* Journal of Nonparametric Statistics. 15, 17-30

[3] Andreatta M, Nielsen M (2016) *Gapped sequence alignment using artificial neural networks: application to the MHC class I system* Bioinformatics. Feb 15;32(4):511-7

[4] Nielsen M, et al. (2003) *Reliable prediction of T-cell epitopes using neural networks with novel sequence representations.* Protein Science. 12:1007-17