Improvement of Machine Learning Algorithms for Predicting MHC-I and Peptide Binding Affinity

Yasmin Heimann

October 2019

Advisor: Raanan Fatal

Abstract

The binding of peptides to MHC class I molecules (MHC-I) is essential for antigen presentation to cytotoxic T-cells, which is a crucial stage in the process of the immune response. Thus, developing an algorithm that accurately predicts the MHC-I and peptides binding affinity is necessary in the fields of immunology and immunotherapy. Nowadays, there exists multiple machine learning algorithms that attempt to predict this binding affinity, with low rate of success. These algorithms use the IEDB database of immune epitopes as a main source for the training data. In our project, we have explored the IEDB database, aiming to analyze any data inconsistencies that might cause an error in the tools available today.

By analyzing the IEDB data, we came up with different data filtering methods to extract new data sets. These data sets were checked using the MHC-Flurry[i] neural network package in Python. We have created an MHC-Flurry learning module that learns a given data and extracts its error rate. The module was used on the full IEDB data and on the new data sets created, for estimating the change each filtering method has given to the learning accuracy.

We found a small improvement in some filters, that points to more specific directions that are worth checking and exploring more deeply. Yet it appears that the lack of massive data in most alleles may be the main reason for the low success rate of the current machine learning tools.

1 Introduction

1.1 Background and Research Question

The immune system is a complex system composed of many processes that activate an immune response. The white blood cells, called T-cells, are responsible for recognition and directly attacking foreign and harmful substances. T-cell immune response is activated by the recognition of an immunogenic peptide antigens bound to the major histocompatibility complex (MHC). These complexes are presented on the membrane of all nucleated body cells as well as by antigen presenting cells, that sample their surroundings, and constitute the essence of the adaptive immunosurveillance mechanism.

This highly specific recognition process consists of multiple selection stages, among which the most selective step is the match between an MHC class I allele and a cleaved peptide, as illustrated in Fig.1.

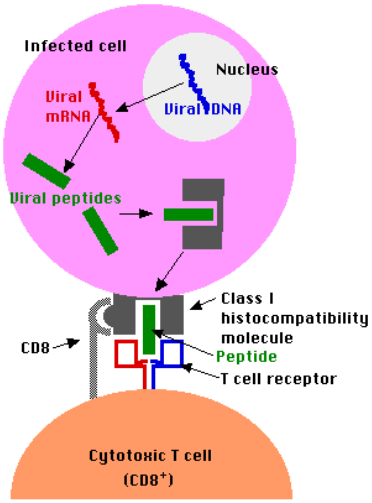


Figure 1.1: An illustration of a cell MHC-I binds to an inner peptide that is recognized by the Cytotoxic T cell of the immune system

From the information above, we understand that binding of peptides to MHC class I molecules (MHC-I) is essential for antigen presentation to cytotoxic T-cells, which is a crucial stage in the process of the immune response. Thus, developing an algorithm that accurately predicts the MHC-I and peptides binding affinity is necessary in the field of immunology and immunotherapy.

These predictions hold the key to the design of effective vaccines as well as highly specific cancer immunotherapies. The recent development of the CAR T-Cells for targeting B-Cell (CD19 bearing) leukemia and lymphoma, marks a recent clinical breakthrough that rises the importance of understanding the mechanisms of cells' recognition by the immune system[ii].
There are roughly 10^{13} combinations in this lock-and-key problem that pose a major challenge in modelling or learning this binding function. Current state-of-the-art computational approaches rely on sophisticated machine-learning models, such as artificial neural networks (for example NetMHCpan3 [iii]and MHC-Flurry[iv]), that require very large training datasets[v]. While the Immune Epitope Database managed to curate tens of thousands of binding assays (giving the values of IC50, EC50 and Kd), this data suffers from various contaminations that we refer to as noise. Different machine learning approaches were implemented on the data available in the IEDB[vi]– Immune Epitope Database – and have shown moderately satisfying results.
We believe that one of the reasons for this non-optimal performance is noise and errors in the curated data.
We would like to know, given the IEDB, what the sources of the error are, and how we can reject strong outliers to reduce this and achieve better performances?

1.2 Research Goals

Our main goal is to evaluate the amount of noise in the IEDB dataset and find solutions for minimizing this noise. For that we want to understand what the sources of the noise are (is it mistyping? different conditions when executing the assays, like temperature?) and estimate its magnitude.
Following that, we would want to create a new dataset, where the outlier assays are filtered, and evaluate machine learning prediction methods over the cleaned dataset versus the original dataset.

1.3 Work Resources and Assumptions

The data used for this research project will be collected from the Immune Epitope Database (http://www.iedb.org), and the assays' original articles. The IEDB dataset contains information on the binding affinity of MHC class I proteins to immune epitopes – the molecular targets of adaptive immune responses – curated from the published literature and submitted by the National Institutes of Health funded epitope discovery efforts (see the IEDB interface and record example in Fig.1.3.1 and Fig.1.3.2)

The affinity values are measured using the nanomolar units and the constants:

1. Kd – equilibrium dissociation constant, evaluates and ranks order strengths of bimolecular interactions. The Kd is a specific type of equilibrium constant that measures the propensity of a larger object to separate (dissociate) reversibly into smaller components.
2. IC50 - half maximal inhibitory concentration, a measure of the potency of a substance in inhibiting a specific biological or biochemical function. IC50 is a quantitative measure that indicates how much of a particular inhibitory substance (e.g. drug) is needed to inhibit, in vitro, a given biological process or biological component by 50%.
3. EC50 - Half maximal effective concentration, the concentration of a drug, antibody or toxicant which induces a response halfway between the baseline and maximum after a specified exposure time.

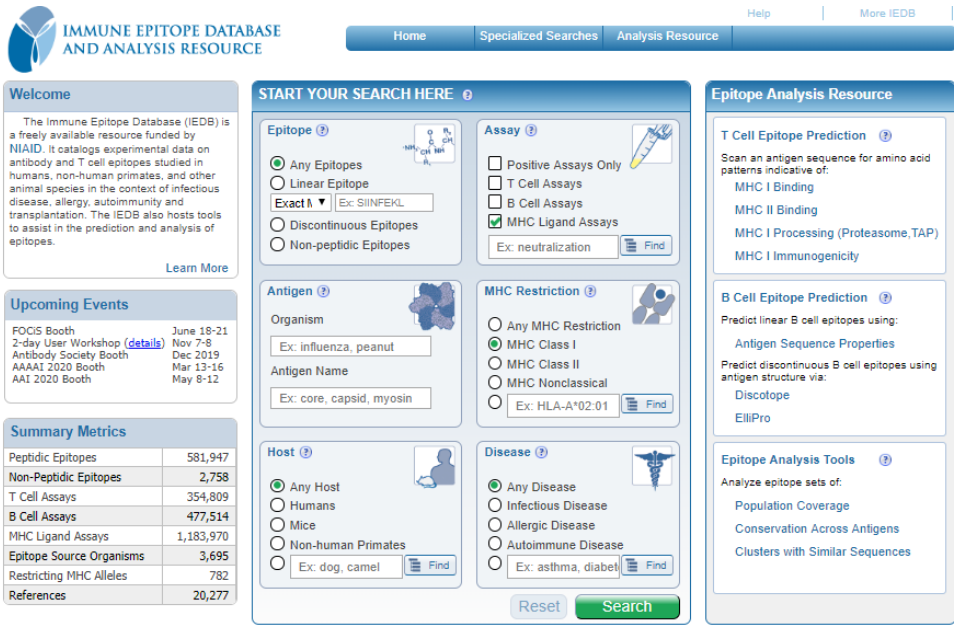


Figure 1.3.1: The interface of the IEDB

ID	Reference	Epitope	Antigen Processing	MHC Restriction	Assay Description	Quantitative Measure
3339081	Massimiliano Baratelli; J Gen Virol 2017	GMIDGWYGY hemagglutinin (360-368) Influenza A virus (A/swine/Spain/SF11131/2007(H1N1))		SLA-1*07:02	binding assay dissociation constant KD **Negative**	= 20000 nM

Figure 1.3.2: The assays' record and data on the IEDB website

2 Results

2.1 IEDB Data Analysis

For the purpose of analyzing the IEDB data and the amount of outliers in it, I have wrote a Python module that uses the 'Pandas' package, which manipulates the data and extracts the relevant assays from it conveniently.

We shall first define and clarify the following terms regarding the data we dealt with:

1. **Duplication** is 2 or more assays in the data that is performed on the same MHC-I and peptide – identical assays. An example from the data:

Reference ID	Epitope ID	Description	Allele Name	Units	alitative Meas	rement Ine	ative measu
1019512	42141	MMLVPLITV	HLA-A*02:01	nM	Positive-High	0	0.5
1000741	42141	MMLVPLITV	HLA-A*02:01	nM	Negative		78100

2. An **error** is defined as the distance between two binding affinity values of each duplicated pair, where the values are normalized in a log scale:

The log scale normalization formula:

$$affinity\ value = \frac{\log\left(affinity\ value + 1\right)}{\log\left(50K\right)}$$

The error of two duplicated assays with binding affinity values v1, v2:

$$error(v1, v2) = abs(v1 - v2) = abs(\frac{\log(v1 + 1)}{\log(50K)} - \frac{\log(v2 + 1)}{\log(50K)})$$

3. An **outlier** is a duplication with high error rate, as defined above. That is because a high contradiction in two identical assays, can affect the ability of machine learning methods to learn from these assays and then apply correct predictions.

2.1.1 Data Extraction

First, I have downloaded as a csv file the relevant training data – which is the IEDB assays including MHC-I and peptides that have binding affinity values.

That was done through the iedb.org, using the following querying categories:

1. Linear Epitope, MHC class I restriction
2. MHC Ligand Assays::binding constant – gives assays with KD, IC50, EC50 and more units.
3. Any host, Any disease.

2.1.2 Data Cleaning and Duplications Analysis

In this stage we cleaned irrelevant assays, evaluated duplicated assays and mapped the potential noise in the data. This was done using computational methods in Python including 'Pandas' package for big data manipulations.

We used the following arrangements on the data before we could use it for the relevant analysis:

1. The data contains inequalities that indicate whether the given affinity value is accurate ("=") or approximately above or under a certain value ("<" or "<"). This feature was added after some assays were inserted to the database, therefore we had blank inequalities which were referred as accurate ("=") in the analysis. Moreover, the approximated values were dropped.
2. Prediction tools support only peptides of length 8-15 amino acids, thus any assay that differs from this convention was dropped.
3. Assays with no binding affinity value were dropped.

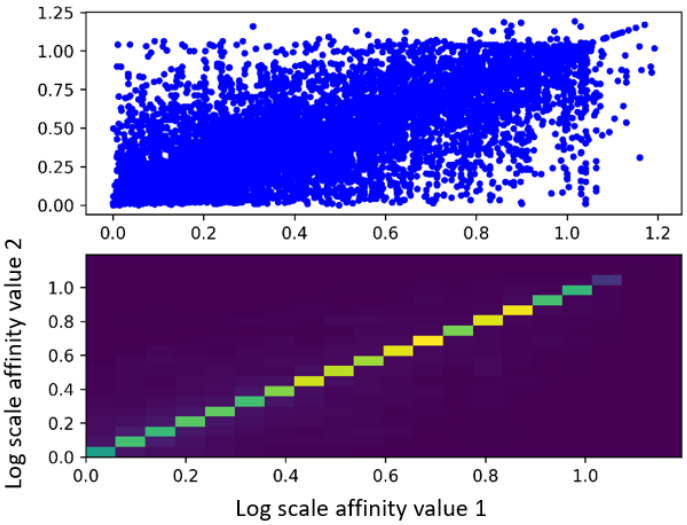
After performing these primarily steps, we get the following number of training assays:

Total assays in the data	~150K
Total assays after cleaning	~102K
Total rows that are part of a duplication	~36K
Total distinct duplications	~17K

2.1.3 Data Error Analysis

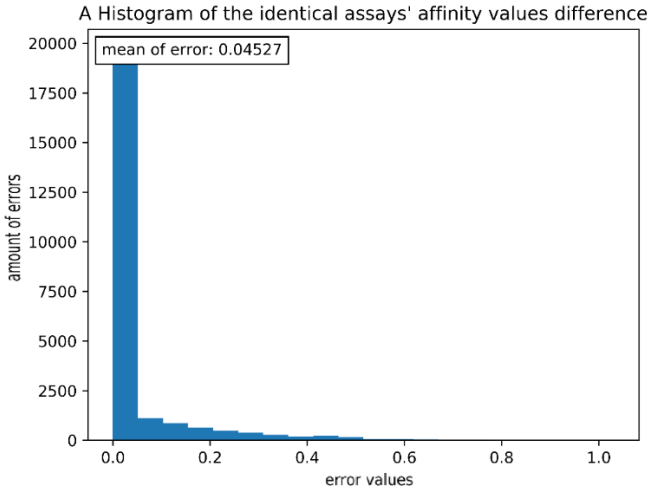
In this stage we analyzed the outliers in the data, by looking at the over-all errors of duplicated assays in the data (as explained in section 2.1). Then, we extracted the duplicated pairs with the greatest errors for further noise analysis and outliers' source detection.

First, we generated a density graph that shows the scattering of the outliers.
The graph presents all pairs of duplications in the data. Given a duplication with v1, v2 - the two normalized binding affinity values, where v1 is the x axis and v2 is the y axis.
The top graph is a scattered graph and the bottom graph is a density graph:



From the scattered graph, we see dots that exceed the diagonal, which indicates error. The density graph implies that most of the duplication have correlation between their affinity values, though some errors do exist, which is satisfying as most of the data is expected to be correlated.

Second, we looked at the error distribution using the following histogram:



Average Error	0.04527
Amount of errors with an error greater than **0.2**	2050
Amount of errors with an error greater than **0.4**	692
Amount of errors with an error greater than **0.6**	169

We see that the average error is low, though some high errors exist, which again is satisfying as most of the data is expected to be correlated, but with some error.

Lastly, we extracted 3 excel files with the 10, 35 and 100 assays' duplications that has given the greatest errors, so we could try and find the source of the noise through them.
Each duplication group in the excel includes all 86 parameters provided from the IEDB, for example:

Error: 7 0.8214 with values: 1.000,0.178

	MHC ligand ID	Reference ID	Epitope ID	Description	Allele Name	Units	alitative Measurement	Inequative measu	PubMed ID	Date
0	205122	200086	7198	CTPYDINQML	Mamu-A1*00	nM	Positive-High	5.9	9637523	1998
1	1183582	1000983	7198	CTPYDINQML	Mamu-A1*00	nM	Negative	50000		2006

2.2 Outliers Analysis and Filtering Methods

After extracting the excel of the greatest errors, we looked at the results in the excel and developed several approached for analyzing the source of these outliers. The outcomes of this chapter will be checked using a machine learning tool:

2.2.1 Exploring the Assays in their Articles

For analyzing the source of the outliers we have detected, we approached the generated excels of greatest errors, and explored them for the aim of understanding what features could lead us to possible outliers.
The duplications' assays found in the excel show, within each set of duplications, a great difference in the affinity values.
When exploring the articles that performed these assays, to try and find the source of the difference, we found minimal or no data on the assays performed. The main difference that we recognized in the articles was different methods and techniques for performing each assay.
When exploring the data in the excel of each duplication individually, we noticed that some duplications with more than one assay can be satisfied by "majority vote" - a common range of affinity values with a smaller sub-group of exceptions from this range.
Also, features of "Meta-Data" – the year that the assay was performed in and the measurement constant (KD, IC50 or EC50) - were suspected to be candidates for an outlier source.
Therefore, we have decided to focus on these parameters and filter the data through them in the ways explained below.

We generated data sets using the following filtering approaches, and each data set was divided into a train and test data set.

2.2.2 Meta-Data

We generated the following data sets:

1. Year (Full) - keep only assays performed after 1995 or 2000, out of all the data.
   The year approach is worth checking based on the assumption that the more up-to-date measurements methods may be more accurate.
2. EC50 (Full) – take out assays measured with EC50 values, out of all the data.
   The EC50 measurement constant approach is worth checking based on the property that EC50 measurements may differ from IC50 and Kd:
   - IC50 is an inhibitory parameter, where EC50 is an activating parameter, and when a peptide binds to a protein it does not mean it will activate it (EC50) – which means not all IC50 values has an equivalent EC50.
   - Kd describes binding, where EC50 describes activity (in the form of effectiveness). Also, while the Kd constant reflects drug concentration that is needed for a level of tissue response, the EC50 constant reflects the drug concentration needed for an amount of receptor binding.
3. Year (Partial) - keep only assays performed after 2000, out of the duplicated data.
4. EC50 (Partial) - take out assays measured with EC50 values, out of the duplicated data.

The number of drops in each filter is as follows:

Year (Full, 1995)	553
Year (Full, 2000)	3721
EC50 (Full)	~40K
Year (Partial)	1413
EC50 (Partial)	4135

2.2.3 Content Approach

We have generated a data set that filters affinity values of duplications by the principle of "Majority Vote".

First, we set a threshold of 500[vii] nM which is routinely used as a binding affinity threshold for peptide selection.
Then, for each duplication group of more than 3 duplications, we keep only values that agree with the majority vote on their affinity value. The majority vote is determined by the threshold of 500nM – if more than 50% of the affinity values are greater than 500nM – only assays with affinity value greater than 500nM are kept, and vice versa.
The number of drops in this filter is 331 assays.

2.3 MHC-Flurry Neural Network - Prediction Tool

For the purpose of checking the improvement of the filtering approaches, we developed in the previous section, we created a prediction tool by building a module in Python that trains a pan-allele neural network and then tests the data.
The neural network we used is the MHC Flurry, which is a well-documented package that allows to train the neural network using our own data.
The MHC Flurry package is an ensemble of neural networks, where each MHC-I allele has its own neural network, that uses as training input all the peptides and their affinity values to the particular MHC-I allele.
We used the following parameters in the run of the network:

1. The train and test data sets were divided into 95% train data and 5% test data. The division was made per MHC-I allele – for each distinct allele, the peptides to train and to test on were divided in a 95% and 5% division.
2. The error rate of the test set was calculated using the MSE and the Pearson correlation.
3. For a “Sanity Check”, we used a data set that is filtered on the highest 10% and 20% errors of the duplications. That is an additional filter applied on the full data, which drops the duplications with highest errors. That is used for validating our assumption that these errors do create data outliers.
4. When training the network, we can choose to apply multiple models for each allele predictor. The outcome of this is that the prediction made by an allele predictor uses 20 models, and then calculate the geometric mean from all these predictions.

The following table shows the resulting train and test losses of the network, when using **one model**. The test was performed over the full test results alleles and over the test results of the 5 and 20 alleles, with the largest amount of peptides data for these alleles.

Filter Type	#Drops	Train error – MSE	Test error – MSE	Test error - Pearson Correlation	Test error – using the largest 5 alleles	Test error – using the largest 20 alleles
Full Data	None	**0.03490431**	**0.037602643**	**0.74110953**	**0.045014178**	**0.039198797**
Highest 10	4527	0.03385406	0.037306209	0.73702425	0.040682601	0.038375419
Highest 20	8176	0.03512287	0.038018878	0.73276623	0.044619267	0.039141603
Year 1995	553	0.036020003	0.038847115	0.73362635	0.044104388	0.040003324
Year 2000	3721	0.03470676	0.036801806	0.75278064	0.043698516	0.038199984
Year (Partial)	1413	0.03565349	0.037685240	0.73826171	0.042097731	0.037981635
EC50 (Full)	40045	0.04419005	0.049002703	0.66572155	0.0564226599	0.051352194
EC50 (Partial)	4135	0.034898003	0.037710073	0.72780755	0.0452407313	0.038945740
MAJORITY VOTE	331	0.035715796	0.039776610	0.71223176	0.0489817516	0.039988529

Figure 2.3.1: *Error results and analysis of the different filtering methods. The "Full Data" is used as a reference, and the yellow marked numbers are the values that are better than the "Full Data" values.*

The following table shows the resulting train and test losses of the network, when using **20 model** over the largest 5 alleles.

Filter Type	#Drops	Train error – 5 largest alleles MSE	Test error - 5 largest alleles MSE
Full Data	None	**0.0297813627**	**0.03308704281**
Highest 10	4527	0.0288269163	0.03230604747
Highest 20	8176	0.0280597984	0.02943374843
Year 1995	553	0.0298078950	0.03053016682
Year 2000	3721	0.0297516124	0.02893257339
Year (Partial)	1413	0.0300077913	0.02996200657
Majority Vote	331	0.0281764225	0.03036870333
EC50 (Partial)	4135	0.0298353803	0.03041409293

Figure 2.3.2: *Error results and analysis of the different filtering methods using 20 models of the greatest 5 alleles. The "Full Data" is used as a reference, and the yellow marked numbers are the values that are better than the "Full Data" values.*

In order to understand what number of alleles will show the best results, we ran the "FULL" data over 20 models, calculating the error rate of alleles with more than 1000 peptides as learning data (~35 alleles in total).

Number of alleles tested	Train MSE	Test MSE
5	0.0297155109415	0.0329745654654
10	0.0289765796506	0.0313587644454
15	0.0280240986696	0.0302340536719
20	0.0274822656314	0.0296314057283
25	0.0271575622782	0.0292777859599
30	0.0273121017139	0.0294492610263
35	0.0273076955079	0.0296585454050

Figure 2.3.3: *Error results of the "FULL" data using 20 models over the greatest 5-35 alleles. The yellow marked numbers are the values that show the best error rate (lowest).*

3 Discussion

Our research project goal was to analyze the IEDB data and attempt to identify problems and conflicts in the data that might have lead to data outliers and thus difficulty in achieving a good accuracy when applying machine learning methods to produce a predictor for MHC allele and peptide binding affinity.

From the results section and the final errors, the filters given by our MHC-Flurry training module we used on the data, we can infer the following:

1. As expected, there is a small improvement in the "Highest" filters. The "Highest10" filter show some improvement, though the "Highest20" filter is very close to the original error. That might be caused by the larger number of drops in the "Highest20" filter.
2. The **"Year"** filters show a slight improvement, which can be a good direction that is worth checking more thoroughly. The "Year2000" even shows some improvement in all the measurements that were performed, which reinforces the idea of focusing on these filters.
3. The **"EC50 FULL"** filter does not show any improvement and has even a significant rise in the error rate. This is reasonable since 40K drops is a significant number of assays to drop, and this can strongly affect the ability to create a successful prediction tool. The **"EC50 Partial"** does not show any significant change from the original data, which indicates that it is probably not an outlier source. These conclusions make sense as the EC50 measurement should be correlated with the Kd and IC50 in most cases.
4. The **"Majority Vote"** does not show any significant change from the original data, when using one model to train and predict. This can lead to the thought we should try to apply more thresholds and try some different majority vote methods, and thus check if there is still some feasibility to this filtering method direction. When using the 20-model parameter, we get improvement of the prediction over the 5 largest alleles, which may indicate that this filter can improve predictions in alleles with massive learning data.
5. As for the **Models** parameter, we observe a significant results improvement when using 20 models, compared to the results of the predictions when using 1 model. That shows the power of this parameter and raises the issue that the smallest alleles (with the least amount of learning data) may affect the results and contribute to high errors.

As shown in table 2.3.3, ~25 largest alleles show the lowest and thus best error rate. From an additional check we made, 20 models showed the best error rate for the network. We can conclude from this section, that the network will perform at its best on the greatest 25 alleles (which have ~1.5K learning samples and more), and when combining training and prediction using 20 models.

Overall, we recognize a small improvement of the prediction, though not a significant one. Moreover, the results seem to have some unresolved issues, such as that the test error is greater than the train error. This leads us to the belief that the results may not be accurate or reliable enough. It can be explained through the challenges we have encountered in extracting the MHC-Flurry predictions and the decisions made in the data filtering phase.

First, the MHC-Flurry uses an ensemble of predictors, which then extract a geometric mean when predicting a new affinity value. When defining the number of models, we chose to use only one model, though we shall consider using more models for having the ensemble effect.

Second, some alleles have a small number of peptides which can lead to overfit or unreliable results due to lack of data. We shall consider dropping alleles with small number of peptides from all data sets, as was partially done in the results section.

We have used various filters, though most of them did not give us any satisfying results. That leads us to think we may need to use a wider approach in the filtering process – divide each filter into sub-filters with more parameters per each filter. For example, in the Year filter, analyze more year thresholds.

As for the testing method, we can consider a 5-fold cross validation which may give us a better sense of the error and may improve our ability to analyze correctly the results we get.

In conclusion, it seems that the lack of massive data is a major weak spot in the machine learning prediction tools, which may be a main reason for the high error performed by all previously published predictors.

3.1 Further Plans

After applying these filtering methods, and after we had a better understanding of the data, we shall continue in the following directions:

1. The Year 2000 Filter has shown some more satisfying results, a fact that leads us to think that we might need to analyze more Year filtering around 1995, as 1996, 1997 etc. and see the impact – for example, what year shows the best results from 1995 until 2000.
2. The "Majority Vote" filter uses the 500nM threshold, which is a common convention, though the threshold may differ depending on the specific allele. Thus, creating the "Majority Vote" filtered data set using various threshold may lead us to new possible conclusions.
3. Combine filters – for example Majority vote and assays performed after 1995.
4. The conditions of an assay (pH, heat etc.) affect the measured value, though it wasn't usually detailed in the articles. Thus, we can choose representative assays with values contradiction (high error) and contact the article's authors to try and understand if there is a difference in the conditions that may have cause this high error.

4 Methods

4.1 MHC Flurry[viii]

MHC Flurry is an open-source package for MHC-I binding affinity prediction. The software implements allele-specific neural networks that use a novel architecture and peptide encoding scheme.

MHC Flurry is freely available to use, retrain, or extend, includes Python library and command line interfaces, may be installed using package managers, and applies software development best practices. The implementation of the package is in Python (versions 2.7 and 3.4+ are supported) using the Keras neural network library.

MHC Flurry is an ensemble of MHC I allele-specific predictors. Separate models are trained for each allele. The input to each model is a peptide of length 8-15 amino acid (others will not be supported). No representation of the MHC allele, such as its amino acid sequence, is used. The models are trained independently, and no information is shared between alleles.

For each allele, MHC Flurry includes an ensemble of eight models. The final nanomolar affinity prediction is taken to be the geometric mean of the individual model outputs. The variance of the

individual model predictions gives an indication of the uncertainty of the prediction and is also made available to users.

4.1.1 Neural Network Architecture

The MHC Flurry predictors are feedforward neural networks composed of the following layers: the peptide representation encoded as a 1-hot (binary) vector, two locally connected layers, a fully connected layer, and a sigmoidal output.

Locally connected layers are one dimensional convolutional layers without weight sharing. Each neuron receives a neighborhood of adjacent points, instead of the full input from the previous layer as in a fully connected layer. The locally connected layers use hyperbolic tangent (tanh) activations, and the fully connected layer uses a rectified linear (ReLU) activation. The weights of the fully connected layer are L1 regularized.

4.2 Error Measurement

4.1.1 Pearson Correlation

The Pearson correlation co-efficient is a measure of the linear correlation between two variables X and Y, and for our purpose - between two data sets.

Its values are between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation, and it is widely used in the sciences.

4.1.2 MSE

The **mean squared error** of an estimator (the prediction tool) measures the average of the squares of the errors - that is, the average squared difference between the estimated values and the actual value.

For each pair of predicted affinity value (Y) and its true value (X) and for n pairs, the formula is used as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2$$

5 References

[i] Nielsen M, Andreatta M. NetMHCpan-3.0; improved prediction of binding to MHC class I molecules integrating information from multiple receptor and peptide length datasets. Genome Med. 2016;8(1):33. Published 2016 Mar 30. doi:10.1186/s13073-016-0288-x

[ii] Comber, Joseph D and Ramila Philip. “MHC class I antigen presentation and implications for developing a new generation of therapeutic vaccines” Therapeutic advances in vaccines vol. 2,3 (2014): 77-89.

[iii] Nielsen M, Andreatta M. NetMHCpan-3.0; improved prediction of binding to MHC class I molecules integrating information from multiple receptor and peptide length datasets. Genome Med. 2016;8(1):33. Published 2016 Mar 30. doi:10.1186/s13073-016-0288-x

[iv] Timothy O'Donnell, Alex Rubinsteyn, Maria Bonsack, Angelika Riemer, Jeff Hammerbacher MHCflurry: open-source class I MHC binding affinity prediction.
bioRxiv 174243; doi: 10.1016/j.cels.2018.05.014

[v] Kim Y, Sidney J, Buus S, Sette A, Nielsen M, Peters B. Dataset size and composition impact the reliability of performance benchmarks for peptide-MHC binding predictions. BMC Bioinformatics. 2014;15(1):241. Published 2014 Jul 14. doi:10.1186/1471-2105-15-241

[vi] Vita R, Overton JA, Greenbaum JA, Ponomarenko J, Clark JD, Cantrell JR, Wheeler DK, Gabbard JL, Hix D, Sette A, Peters B. The immune epitope database (IEDB) 3.0. Nucleic Acids Res. 2014 Oct 9. pii: gku938. PubMed PMID: 25300482.

[vii] Paul S, Weiskopf D, Angelo MA, Sidney J, Peters B, Sette A. HLA class I alleles are associated with peptide-binding repertoires of different size, affinity, and immunogenicity. J Immunol. 2013;191(12):5831–5839. doi:10.4049/jimmunol.1302101

[viii] Timothy O'Donnell, Alex Rubinsteyn, Maria Bonsack, Angelika Riemer, Jeff Hammerbacher MHCflurry: open-source class I MHC binding affinity prediction.
bioRxiv 174243; doi: 10.1016/j.cels.2018.05.014