TECHNICAL UNIVERSITY OF DENMARK

21st JUNE 2022

22125 - ALGORITHMS IN BIOINFORMATICS

Influence of allele size on the prediction power of PSSM, SMM, and ANN to determine MHC-binding peptides

Group 7

Adikrishna Murali Mohan (s212940)

Prince Ravi Leow (s220520)

Baris Kara (s213449)

Abstract

Peptides presented by the major histocompatibility complex class I (MHC-I) molecules play an important role in cellular recognition and immunological response triggered by T-cell lymphocytes. [1] In this study, 35 different Human MHC-I alleles were used to evaluate the influence of different allele sizes, on the predictive power of Position Specific Scoring Matrix (PSSM), Stabilization Matrix Method (SMM) gradient descent and Monte Carlo, and Artificial Neural Network (ANN) algorithms. We found that in general, SMM (gradient descent implementation) and ANN yielded the best and most consistent results, throughout all alleles (ANN having the best performance overall). However, it is also worth noting that ANN is significantly more computationally expensive (~14-fold higher execution time than gradient descent). We also found that the number of binders appears to be a more consistent predictor of performance. Due to time constraints, a number of good practices could not be implemented - such as true nested cross-validation, accounting for data redundancy, and hyperparameter optimisation. For future studies, it would be desirable to prioritize these, to ensure optimal model quality.

Introduction

MHCs play a major role in the recognition of indigenous and exogenous cells which trigger adaptive immune responses against pathological cells. The MHC molecules are cell-surface proteins that bind to a specific set of protein sequences (peptides) on the outer cell membrane to trigger cellular recognition by cytotoxic T-cell lymphocytes. [2] Due to this mechanism nature, the characterization of peptides that can bind to specific MHCs plays an important role in the identification of new epitopes for pharmaceutical applications such as tumor recognition and vaccine development. The binding affinity of a given peptide to a specific MHC molecule is mainly determined by the amino acid composition of the binding core. Hence, this enables benchmarking of MHC binding peptides by *in silico* methods. [3-5]

In this study, benchmarking of MHC-I binding peptides is tested by three algorithms: PSSM, SMM, and ANN. The PSSM algorithm is a weight matrix that illustrates the likelihood of observing a specific amino acid at a given sequence position. However, PSSM lacks the flexibility to capture a complete alignment for large polypeptides. [6] Fortunately, this problem can be solved by SMM algorithms such as Metropolis Monte Carlo & gradient descent which can be used to optimize the evaluation performance by determining the global minima for a given function. However, the SMM algorithms lack the power for benchmarking poly-specific MHC-I binding motifs. [7] This shortcoming can be solved by ANN, where multi-layer neural connections are applied to capture higher-order interactions. [1] Nevertheless, this advantage comes at the cost of computational power/execution time. In conclusion, the prediction performance and computational cost of algorithms show alteration depending on the characteristics of the dataset. Hence, the determination of the right algorithm to be applied must be achieved by making a trade-off between the execution time and dataset characteristics.

Materials & Methods

Data

The dataset used for the comparative study is obtained from the research paper by Peter et al. covering MHC molecules. [1] The data describes peptide sequences for 35 alleles, partitioned into 9-mer chains, with their respective binding affinities. In general, the number of binding sites generally increases with allele size (Figure 1). However, the allele size distribution in the dataset is not uniform. For example, allele A0201 contains 1181 binders - 1.7 times more than the second most binder-abundant allele, with an allele size of 3089 amino acids - 1.5 times the allele size of the second largest allele. In addition, there are

alleles with a very low amount of binders relative to their size - for example, allele B4001 - with an allele size of 1078 with only 40 binding sites. Therefore, the number of binding sites, relative to the allele size will be taken into account, during the evaluation.

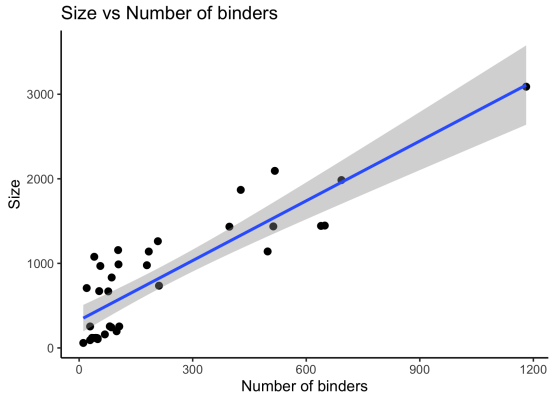

Figure 1: *Shows the correlation between the allele size and the number of binders.*

Evaluation of the models are conducted via cross-validation. The data are divided into 5 equal pieces, where 4 are used for training, and 1 for subsequent model testing and performance evaluation. This constitutes the so-called 'un-nested' cross-validation method, and is used for all methods, except for PSSM, in which the evaluation is performed directly on the test-set. The ideal solution would be to use 'nested' cross-validation, in which one partition was isolated from the training and testing process, as to not introduce any biases, and thus develop an overly optimistic prediction model. However, this would naturally introduce one extra step, and due to time constraints, this had to be cut from the process.

Another factor that can impact the results is data redundancy - which could lead to over-generalisation, due to overabundance of certain amino acid residues. Normally, this would be accounted for via heuristics (e.g. a Hobohm sorting algorithm). However, due to similar time constraints, this was not possible to implement, and thus amino acid redundancies are not accounted for in this study.

Methods

PSSM

PPSM uses sequence weighting and pseudo count correcting to generate a weight matrix, which is then normalized by the background frequency of amino acids to generate a score that represents the partial frequency of observing a specific amino acid at a given position of 9-mer sequence.

Algorithm design:

In the first step of the algorithm, sequence alignment of 9-mers is used to generate a scoring matrix, which is followed by the clustering of redundant 9-mer sequences to avoid bias introduced by highly similar sequences. The resulting alignment is then used to compute the frequency of observing a specific amino acid for a given sequence position. In this step, the frequency of amino acids inside clusters is normalized by the number of sequences located per cluster. Hence, the frequency scores are adjusted by an alpha parameter to base the frequency scores only on the effective number of sequences.

In the next step, the BLOSUM50 matrix is used to calculate pseudo count probability which takes into account the likelihood of substituting other amino acids at a given sequence position. In order to gain more confidence, the pseudo count is optimized by using a weight parameter (beta), which is tested in the range of 50-200 in our report.

Following this step, the adjusted amino acid frequency and weighted pseudo count are used to calculate the partial probability of observing a specific amino acid at a given sequence position. In the final step, calculated partial probabilities per amino acid are normalized by using the amino acid background frequency to take into account the natural abundance of given amino acids. Hence, this results in a PSSM, where each cell shows the generated weight score for observing a specific amino acid at a given sequence position.

SMM

Stabilization Matrix Method (SMM) uses a cost function in order to minimize the error by reducing the distance between the target and predicted values. However, to prevent the overfitting of the model, the SMM algorithm introduces a hyperparameter to adjust the weight of each data parameter. Following this approach, the SMM algorithm optimizes the evaluation performance of a model by determining model parameters that give minimal error.

Algorithm design:

At the initial step, the 9-mer data is set and is separated into 5 equal sizes to form 4 training and 1 test/evaluation dataset as described in section Data. In the next step, a sparse encoding scheme is applied to represent the amino acid sequences in binary numbers. After this step, the test set is trained on the training set which is used as target values to generate the prediction.

At this step, the error function is applied to compute the distance between the target and prediction values. The resulting distance values are used to minimize the error by forcing the prediction model fit to the target values. However, this causes overfitting, thus this step is followed with the addition of a second error term which uses a hyperparameter to optimize the weights on the training set parameters. Hence, we tested different hyperparameter values in the range of 0.01-10 to find the optimal parameter value. After testing, the optimal hyperparameter is divided by the number of target values to obtain the final cost function which computes the error per target variable.

In the next step, the gradient descent algorithm is tested, where the derivative of the final cost function is used to compute the variation in the weight of model parameters. In order to carry out this calculation, a learning parameter (epsilon) is introduced to determine the magnitude of change in the weights. After this calculation, the weight of model parameters is adjusted by subtracting the change in weights. This entire process is then iterated until the weight parameters do not vary which indicates that the global minimum is reached.

If we replace the gradient descent with the Monte Carlo algorithm, the determination of the global minimum is achieved by the optimization of the fitness function. In this approach, the sequence alignments are randomly moved to generate different 9-mer sequences. After this step, newly generated 9-mer sequences are aligned and the energy difference observed between the two 9-mer configurations is computed as the fitness function. In the next step, the scalar parameter (T) is applied, which is gradually decreased throughout the simulation in order to force the model into a high fitness state, and thus, minimum global error. Finally, the probability of accepting the move in the training 9-mer dataset is calculated by taking the exponential of the final step function. In this process, all the moves resulting in higher fitness were always accepted, whereas only a fraction of the moves resulting in lower fitness were accepted due to the probability function.

ANN

Artificial neural network (ANN) uses the process of the biological nervous system to develop algorithms that are used to model complex patterns and predict the data. It has non linear correlation. The structure of ANN consists of an input layer, hidden layer, and output layer. The activity of the input layer represents the raw information put into the network. Hidden layers determine the activities of the input unit and weights on the connection between the input and the hidden layer. The output layer depends on the activity of the hidden layer and the associated weights.

Algorithm design:

In this project, the purpose of the program is to predict the MHC binding peptides. First, we implement the training algorithm. ANN is trained using known data. In the training process, the weights are initialized to small, random values. The objective of the training process is to modify the weights to produce output for different data. This creates a

synapse file which is fed to the forward algorithm.

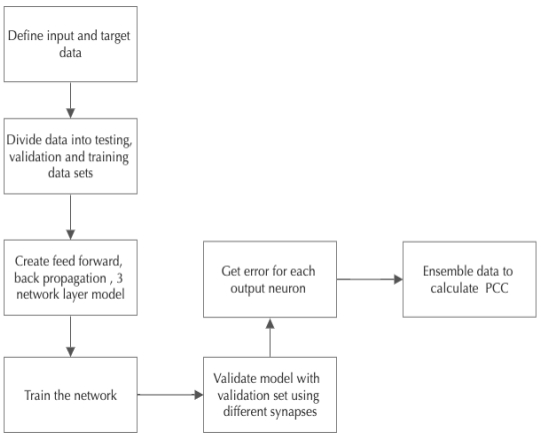

Figure 2: *Flow diagram showing the program pipeline of the ANN algorithm*

The data encodes the peptides using sparse encoding or BLOSUM encoding and then uses a feedforward algorithm to make predictions for the peptides. Amino acids must be numeric scores, in order to input it to the neural network. Sparse encoding converts amino acids to a binary vector, whereas BLOSUM encoding uses BLOSUM50 substitution matrix accounting for the similarity between AA sequences. We trained the network with both BLOSUM and sparse encoding and combined the prediction of both approaches. Analysis was done with default values of epoch 100, epsilon (learning rate) 0.05.

In the first step, the input data is passed to the hidden layer with the weights. We have set the number of hidden layers to 3. Each hidden layer has neurons and in our case, we set it to 5 and the hidden layer performs computation on the inputs. This is done by forward propagation, in which all the inputs are multiplied with their weights. The weight is the coefficient gradient and after the weights are assigned, a bias variable is added to the input. This bias is set to 1 and it helps to find the best fit for the model.

In the second step, the activation function is applied to the input value and then sent to the next hidden layer. This activation function is used to set nonlinearity of the model. After passing all the hidden layers the output goes to the output layer. This gives the prediction value.

After getting the prediction value from the output layer, we calculate the error which is the difference between the actual and predicted value. If the error is large we minimize it by back propagation which is the process of updating and optimizing the weights. We used gradient descent as a backward propagation optimiser.

Gradient descent minimizes the error to achieve a global minimum. This is done by first initializing the weights randomly and finding the derivative of error for new weights. The new weight has a learning rate as a hyperparameter to control the steps in back propagation. This step continues till we get a global minimum and a minimal loss.

It can be noted that the number of hidden layers should be low. This is required to ensure that the ANN does not store all information from the learning set, instead generalizes it to avoid overfitting. The higher number of hidden neurons in the ANN allows detection of higher order correlation, however, it increases the number of hyperparameters.

Hyperparameter optimization

As the algorithms' performance are partially dependent on the hyperparameter input values, these were optimized, to ensure the highest PCC values possible. All subsequent presentation of results will therefore use the optimized parameters, unless stated otherwise.

PSSM

Performance for beta values 50, 70, 100, 150, and 200 were evaluated. There was no significant difference in performance. Therefore, results for beta = 100 were arbitrarily selected for performance comparison.

SMM: Gradient descent

Performance values for λ values 0.01, 0.1, 1, 5, and 10 were evaluated. λ = 1 had the highest average performance and was therefore selected for performance comparison. The effect of the number of iterations (epochs) was also found to be negligible, and therefore the originally arbitrarily selected value of 100 was selected.

Epsilon (learning rate) would have been considered, but due to time constraints, this was not optimized and has therefore been fixed at 0.05 for the duration of this study.

Early stopping has also been employed to ensure maximum performance.

SMM: Monte Carlo

Due to time constraints, and being computationally expensive, relative to its poor performance, hyper parameters were not optimized for Monte Carlo. Therefore, the following parameters were fixed for the duration of this study: λ = 0.01, epochs = 1000, T_i (initial temperature) = 0.01, T_f (final temperature) = 10^{-6}, T_{steps} = 10. Early stopping has also been employed to ensure maximum performance.

ANN

Performance between using BLOSUM and sparse encoding was evaluated. The performance for BLOSUM was significantly higher than sparse encoding, and has therefore been selected for performance comparison.

Due to time constraints, other hyper parameters have been fixed for the duration of the study: Number of hidden neurons = 5, number of hidden layers = 3, epsilon = 0.05, epochs = 100.

Early stopping has also been employed to ensure maximum performance.

Results

In order to quantify predictive power of the algorithms, the Pearson Correlation Coefficient (PCC) is used. Here, we will compare the performance (PCC) relative to the allele size, and their corresponding number of binders. We will also take the computational cost (execution times), relative to performance into account.

Observed performance

PSSM performance

The prediction results obtained from the PSSM algorithm showed that both size and number of binders are positively correlated with the prediction power as illustrated in Figure 3 and 4. In general, increasing the two parameters will result in improved prediction power. However, it can also be observed that within a specific threshold (allele size < 700 & binder number < 300), the algorithm fails to generate a reliable prediction as indicated by the data points observed along with all the PCC values. Overall, the number of binders has a more clear correlation with the PCC values compared to the allele size, thus indicating that the number of binders has more influence on the improvement of prediction power in the PSSM algorithm.

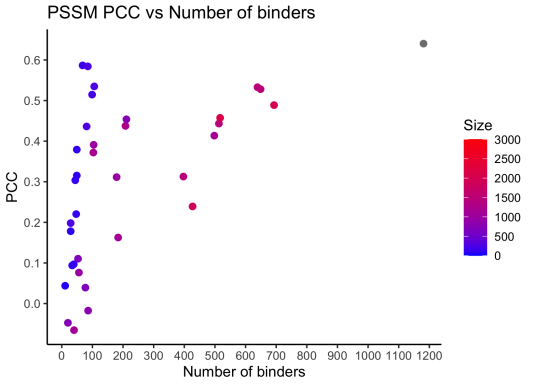

Figure 3: *Shows the influence of the number of binders on the prediction power of the PSSM algorithm, which is quantified by the Pearson Correlation Coefficient (PCC). The colour gradient is used to illustrate the number of binders per given data point (allele).*

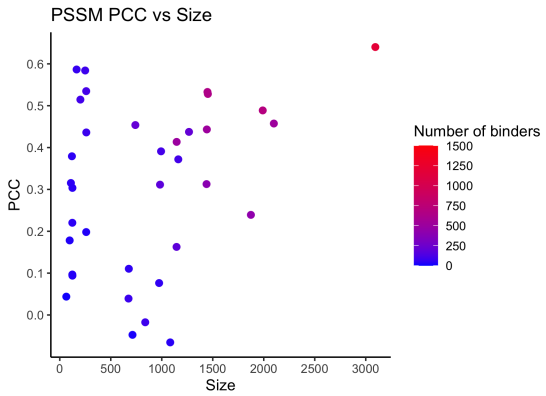

Figure 4: *Shows the influence of allele size on the prediction power of the PSSM algorithm, which is quantified by PCC.*

SMM performance

Due to its middling performance relative to high computational cost, we have chosen not to include the Monte Carlo PCC vs number of binders and size charts in this report. However, in the case of the gradient descent algorithm, both the number of binders and the allele size have a positive correlation on the predictive power of the algorithm beyond the specified threshold values. As being different from the PSSM algorithm, the gradient descent results provide a more evident trend between the tested parameters and the prediction power, as well as an improved peak PCC value (~0.75) as illustrated in Figures 5 and 6 below.

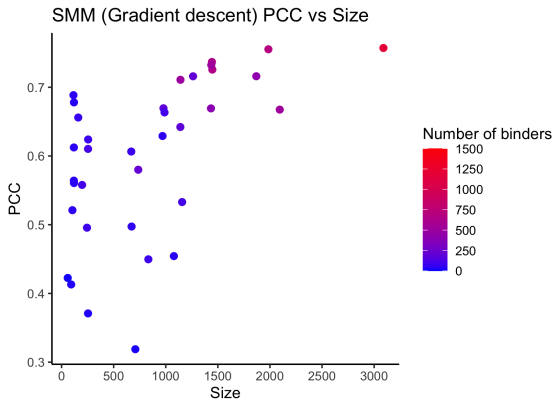

Figure 5: *Shows the influence of the allele size on the prediction power of the gradient descent algorithm, which is quantified by PCC.*

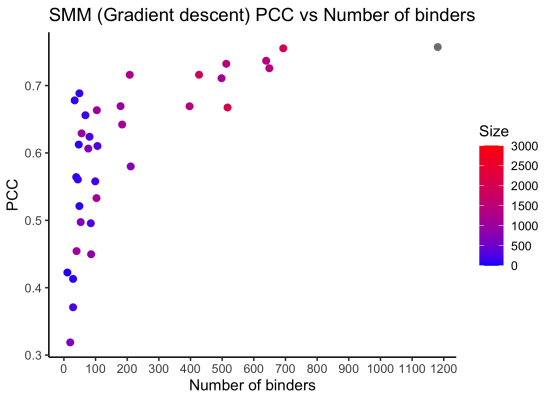

Figure 6: *Shows the influence of the number of binders on the prediction power of the gradient descent algorithm, which is quantified by PCC.*

ANN performance

Figures 7 and 8 show the influence of sample size and the number of binders on the prediction power of the ANN algorithm against PCC, respectively. ANN has the highest peak performance among all the 4 methods - approaching ~ 0.9.

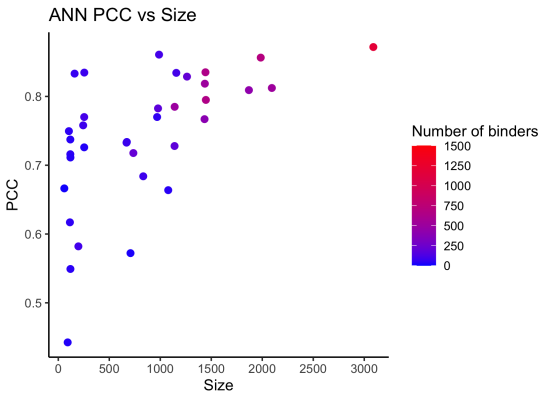

Figure 7: *Shows the influence of the allele size on the prediction power of the ANN algorithm, which is quantified by PCC.*

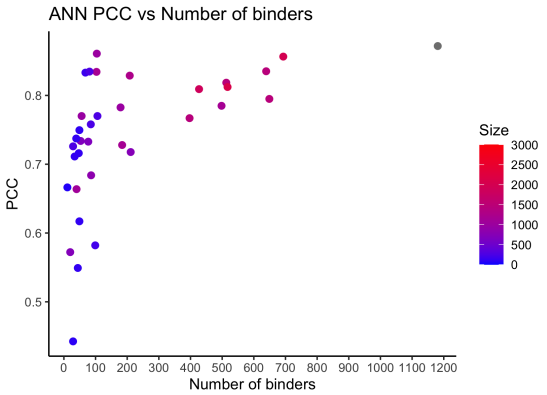

Figure 8: *Shows the influence of the number of binders on the prediction power of the ANN algorithm, which is quantified by the PCC.*

Comparison of PSSM, SMM, and ANN

Superficially, it is observed that the highest performances come from the largest datasets. However, there are instances where the two appear to not be directly correlated. For instance, in PSSM (Figure 4), direct performance correlation with size is unclear. On the other hand, the number of binders appears to be a more consistent predictor of performance. This effect is especially pronounced in Figures 6 and 8, indicating that a critical threshold ~300 binders must be met, before performance is maintained consistently throughout different numbers of binders between alleles. The composite graph (Figure 9) further iterates this point, in which it is observed that for all methods, a critical number of binders is required, before performance consistency is reached.

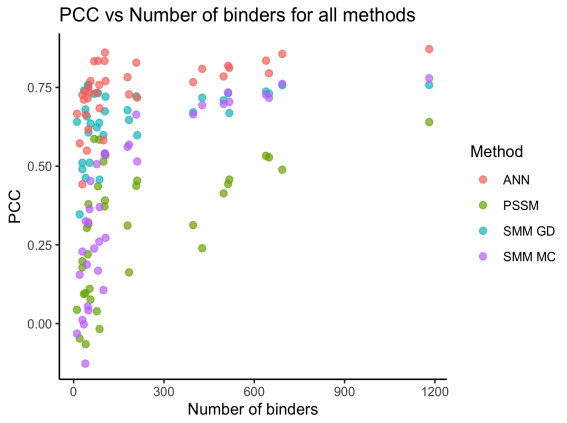

Figure 9: *Shows the predictive power of different algorithms that are quantified by the overall mean PCC values generated versus the number of binders in each allele.*

For this reason, the effect of the number of binders in a dataset is more thoroughly considered. Indeed, as pictured in Figure 1, most (26) of the alleles fall below the so-called critical threshold of ~300 binders. Some (8) of them fall in the middle - at around 300-700 binders, and the single largest allele (A0201) sits at ~1200 binders. As such, method performance will be evaluated, by separating the number of binders into categories, ‘small’, ‘medium’, and ‘max’ - with an overall mean PCC taken as an average of all PCC values for the method.

Displayed in Table 1 are the mean PCC values for all 4 assessed methods. The maximum PCC values are all derived from the performance of the same allele (A0201), and as such, are all directly comparable to each other. A0201 also contains the highest number of binders (1181). At 1181 binders, it has 1.7 times the amount of the second most binder abundant allele in the dataset (A1101), which indicates a direct algorithm performance with number binders.

Table 1: *Summary showing the influence of different allele sizes of binders on the prediction power of the tested algorithms which is quantified by the PCC value.*

Method	Mean PCC (Small | binders < 300)	Mean PCC (Medium | 300 < binders < 900)	Mean PCC (Max | ~1200 binders)	Mean PCC (All)
PSSM	0.258	0.427	0.640	0.308
SMM Gradient descent	0.621	0.715	0.759	0.647
SMM Monte Carlo	0.280	0.712	0.779	0.393
ANN	0.715	0.810	0.872	0.741

In general, ANN and SMM Gradient descent are the highest overall performers, with an overall average PCC of 0.741 and 0.647 respectively, whereas PSSM and SMM Monte Carlo are the poorest performers, both having average PCC values far below 0.5.

The amount of binders significantly affects PSSM, where the average performance in the ‘small’ category is almost 2.5 times lower than its maximum performance. This effect is even more pronounced in Monte Carlo, where it performs moderately in the ‘medium’ and ‘max’

categories, but extremely poorly in the ‘small’ category. On the other hand, gradient descent and ANN’s performances are much less affected by the number of binders.

It is worth noting that while both SMM methods have similar maximum performance, gradient descent’s maximum performance is much closer to its overall mean, than for Monte Carlo. PSSM has the lowest mean and maximum PCC, making it the overall worst performer, in both categories. ANN on the other hand has the highest mean performance and even manages to approach a maximum PCC of ~0.9.

In order to better illustrate the overall performance, a density plot for all PCC values is displayed in Figure 10. It can be observed that both ANN and gradient descent (SMM GD) achieve the highest PCC densities, with more pronounced peaks - indicating a more consistent performance throughout the dataset. PSSM appears to have a median PCC of ~0.45 but drops rapidly past this point, and its density reaches a relative density of 2 - far lower, and therefore less consistent than that of gradient descent and ANN. Monte Carlo’s performance is the most evenly distributed. However, having the lowest peak, and in spite of its moderate performance in the ‘medium’ category of binders (Table 2), it appears to be the least consistently performing between the 4 methods.

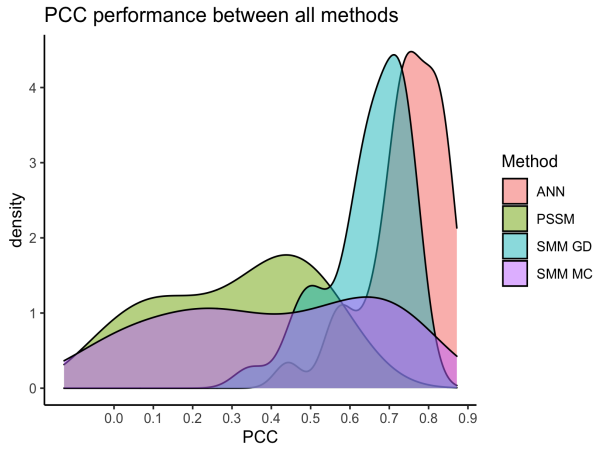

Figure 10: *Density plot for PCC values between all tested algorithms. Shows the comparison between tested algorithms based on the dataset density distribution.*

Computational cost

Considering the performance (Table 1) relative to computational cost (Table 2), SMM provides the best performance-to-cost ratio, given the fact that SMM - Gradient descent provides significantly lower execution time with comparable mean PCC value when compared to ANN, which is the best performing model in the project.

Table 2: *Overview of approximate execution times for all 4 methods*

Method	PSSM	SMM GD	SMM MC	ANN
Approx. Execution time	<5 mins	~30 mins	~1.2 hours	~7 hours

Negative PCC values

It is worth noting that we observed three negative values in PSSM and SMM - Monte Carlo from 3 different sets of alleles in each case. In PSSM the alleles A6901 (size = 838) , B0801 (size = 713), B4001 (size = 1083) gave PCC results of -0.0175, -0.0475, -0.0655 respectively. In SMM Monte Carlo the alleles B4002 (size = 118), B4403 (size = 119), B5701 (size = 59) give PCC values of -0.1271, -0.00242 and -0.0316 respectively. This is consistent with the fact that both methods are the lowest performers, and have a significant overall contribution to the observed mean performance Table 1.

Discussion

The results from the analysis indicate that among the methods: Position Specific Score Matrix (PSSM), Stabilized Matrix Method (SMM) - Gradient Descent, Stabilized Matrix Method (SMM) - Monte Carlo, and Artificial Neural Network (ANN); the ANN is the best-performing algorithm, however, SMM - gradient descent has an overall better prediction power relative to its lower computational cost and time. These results should be taken into account considering a dataset of 35 peptides. Searching with PSSM

and SMM in complete genomes or large sequence databases is computationally expensive as the matrix based methods do not explicitly optimize the cost function. All of the primers used in the analysis are composed of short peptides 9mers, which is mainly due to working with MHC-I alleles that have the characteristic of having short peptide lengths, unlike MHC-II alleles which can have different peptide lengths.

The weight matrix describing a sequence motif can be calculated from a set of peptides of equal length. This approach is appropriate when dealing with MHC class I binding, where the length of the binding peptides is relatively uniform. MHC class II molecules, on the other hand, can bind peptides of very different lengths, and the weight-matrix methods described up to now are hence not directly applicable to characterize this type of motif. In general, this indicates that both a low allele size and a low number of binders relative to allele size can negatively impact an algorithm's performance.

It is worth noting that the negative correlations for PSSM occurred in ‘medium’ sized data whereas Monte Carlo’s came from ‘small’ sized data. This is in general consistent with the rest of the results, as Monte Carlo generally seems to perform poorly on small datasets - with 40% of the alleles having a PCC < 0.3 - all of them having allele size ≤ 255. In the case of PSSM, all of these alleles had 86 or fewer binders - the median value for the number of binders in this dataset. According to the overlapping density graph under the overall performance result section (Figure 10), it can be seen that the critical number of binders is around 300 and thus the number of binders is a clear deciding factor of performance as opposed to the allele size.

Neural networks have certain drawbacks. Sometimes neural network results can yield lower performance as a small change in weight can lead to a significant change in output, data can be overfitting, and time complexity can be high. The algorithms may run for days even for small data sets. When we have large data then the neural network generalizes well otherwise it will overfit the data.

To improve the performance of matrix based methods and artificial neural networks in particular do more hyperparameter optimization, use nested cross-validation and perform data redundancy. For a relatively large dataset, the data should be rationally sub-sampled, to reduce model overfitting. Standardizing the input data can make the training faster and reduce the local minima. In order to improve generalization on small noisy data, train multiple neural networks.

References

1. Peters, Bjoern et al, “A community resource benchmarking predictions of peptide binding to MHC-I molecules”, *PLoS computational biology* vol. 2,6 (2006), doi:10.1371/journal.pcbi.0020065.

2. Janeway CA Jr, Travers P, Walport M, et al. “Immunobiology: The Immune System in Health and Disease - The major histocompatibility complex and its functions”, 5th edition, New York: *Garland Science* (2001).

3. Nielsen, M., Lund, O. “NN-align - An artificial neural network-based alignment algorithm for MHC class II peptide binding prediction”, *BMC Bioinformatics* 10, 296 (2009), https://doi.org/10.1186/1471-2105-10-296.

4. Rao, Arjun A et al. “ProTECT-Prediction of T-Cell Epitopes for Cancer Therapy”, *Frontiers in immunology* vol. 11 (2020), doi:10.3389/fimmu.2020.483296.

5. Usman Sayeed, Gulshan Wadhwa, et al. “MHC binding peptides for designing of vaccines against Japanese encephalitis virus: A computational approach”, *Saudi Journal of Biological Sciences* vol. 25 (2018) pp 1546-1551, https://doi.org/10.1016/j.sjbs.2016.01.041.

6. Ahmad, Shandar, and Akinori Sarai. “PSSM-based prediction of DNA binding sites in proteins”, *BMC bioinformatics* vol. 6 33 (2005), doi:10.1186/1471-2105-6-33.

7. Nielsen M, Lundegaard C, Lund O. “Prediction of MHC class II binding affinity using SMM-align - a novel stabilization matrix alignment method”, *BMC Bioinformatics* (2007), https://doi.org/10.1186/1471-2105-8-238.