Mutating Peptides binding to MHC class-I molecules with Reinforcement learning

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

Foreign peptides bound to major histocompatibility complex (MHC) class-I molecules and presented on the cell surface play a vital role in the immunotherapy. These foreign peptides can be recognized by T-cell receptors to trigger the adaptive immune response. In this paper, we formulate the identification of these foreign peptides as a reinforcement learning (RL) problem, and present a framework, which we call `PepPPO`, to generate peptides that can be presented by any given MHC molecules. `PepPPO` learns to mutate any random peptides into positive peptides that can be presented on the cell surface by the given MHC molecule. The experiments show that our method can successfully generate positive peptides with high presentation scores predicted by the well-developed computational tools. The generated peptides also have been validated by the physical models to be able to bind with the MHC molecules.

1 Introduction

We summarize our contributions below:

1) We formulated the identification of positive peptides as a novel reinforcement learning problem. We built an RL environment from the well-developed computational tools and learned a policy to generate positive peptides within the built environment.

2) Our method can generate positive peptides with large predicted presentation scores. The generated positive peptides are also validated to be able to bind with the MHC molecules by the physical models.

3) We built a database of potentially positive peptides for all the known alleles using our method to accelerate the development of peptide vaccines.

4) We developed multiple different baseline methods and demonstrated the effectiveness of `PepPPO` with a detailed comparison.

2 Related Works

To the best of our knowledge, Angermueller *et al.* [1] first proposed a model-based reinforcement learning method based on proximal-policy optimization (PPO) named `DyNA PPO`, for the biological sequence design. `DyNA PPO` and [10]

3 Methods

3.1 Problem Definition

In this paper, we focus on the generation of peptides presented on the cell surface by the MHC class I molecules. We use a computational tool named `MHCflurry2.0` [6], to estimate the presentation scores of peptides with an given MHC molecule. The presentation score is a composite score of the antigen processing (AP) prediction and the binding affinity (BA) prediction. The AP prediction predicts the probability for a peptide to be delivered by the transporter associated with antigen processing (TAP) protein complex into the endoplasmic reticulum (ER), where the peptide can bind to MHC molecules. The BA prediction predicts the binding strength between the peptide and MHC molecules. Higher presentation scores require higher AP and BA scores, and indicate higher probabilities for generated peptides to be presented on the cell surface by the given MHC molecules.

Problem Definition: We represent a peptide as a sequence of amino acids $\{a_i\}^l$, where a is one of 20 types of natural amino acids and l is the length of the sequence. Given an MHC molecule m, `PepPPO` aims to generate a peptide p with l amino acids that can maximize the presentation scores $r(p, m)$ predicted by the `MHCflurry2.0`.

3.2 Peptide Mutation Environment

The peptide mutation environment consists of 3 components including the state space, the action space and the reward design. In this section, we discuss these 3 components of the peptide mutation environment as below.

State Space: We define the state of the environment s_t at time step t as a pair of an MHC class I molecule and a peptide (p, m). We represent an MHC molecule as a pseudo sequence with 34 amino acids, each of which is in potential contact with the bound peptide within a distance of 4.0 Å, following the previous works for peptide-MHC binding prediction [5, 3]. With a peptide of length l and an MHC molecule, we represent the state s_t as a tuple $s^t = (E^p, E^m)$, in which E^p and E^m are the encoding matrices of the peptide and the MHC molecule, respectively.

The details of the encoding method will be described in Section **??**. During the training, to initialize the state s^0, we randomly sample an MHC class I molecule, and get a random initial peptide sequence with 50% chance of randomly sampling from peptides in the IEDB dataset or with 50% of randomly generating a new one. We define the terminal state s_T as the state with the maximum time step T or with the presentation score greater than threshold σ.

Action Space: We define a multi-discrete action space to optimize the peptide by replacing one amino acid with another one. At time step t, given a peptide $p^t = \{a_i\}^l$, the action is to first determine the position of the amino acid to be replaced, and then predict the type of new amino acid at the position.

Reward Design: We only use the final rewards to guide the optimization of RL agents, that is, only terminal states can receive rewards from the peptide mutation environment. We define the final rewards as the predicted presentation scores $r(p, m)$ of `MHCflurry2.0` between the peptides p_T and the MHC molecules m within the terminal state s_T.

3.3 Pan-allele Mutation Policy Network

In this section, we introduce our policy network learned by the RL agent to mutate the amino acids within the peptides given an MHC molecule. Specifically, the policy network takes the peptide and the given MHC molecule as input and learns to mutate one amino acid in the peptide sequences at each step to maximize the presentation scores.

Encoding of Amino Acids We use a mixture of multiple encoding methods to represent the amino acids within the peptide sequences and the MHC molecules. We represent each amino acid by concatenating the encoding vectors $\mathbf{e}^B$, $\mathbf{e}^O$ and $\mathbf{e}^D$ from the BLOSUM matrix, the one-hot matrix and the learnable embedding matrix, respectively, that is, $\mathbf{e} = \mathbf{e}^B \oplus \mathbf{e}^O \oplus \mathbf{e}^D$ and $\mathbf{e} \in \mathbb{R}^d$. This method has been demonstrated in [2] to achieve the best prediction performance on peptide-MHC binding prediction among all the combinations of these encoding methods. The encoding matrices E^p and E^m of the peptide p and the MHC molecule m are then represented as $E^p = \{\mathbf{e}_1, ..., \mathbf{e}_l\} \in \mathbb{R}^{l \times d}$ and $E^m = \{\mathbf{e}_1, ..., \mathbf{e}_{34}\} \in \mathbb{R}^{34 \times d}$, respectively.

Embedding of States In order to predict the mutation of amino acids in peptide sequences, we first embed each amino acid a_i within the peptide sequences $\{a_i\}^l$ into an continuous latent vector $\mathbf{h}_i$ using one-layer bidirectional LSTM as below,

$$\begin{aligned} \overrightarrow{\mathbf{h}}_i, \overrightarrow{\mathbf{c}}_i &= \text{LSTM}(\mathbf{e}_i, \overrightarrow{\mathbf{h}}_{i-1}, \overrightarrow{\mathbf{c}}_{i-1}; \overrightarrow{\theta}^p) \\ \overleftarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{c}}_i &= \text{LSTM}(\mathbf{e}_i, \overleftarrow{\mathbf{h}}_{i+1}, \overleftarrow{\mathbf{c}}_{i+1}; \overleftarrow{\theta}^p) \\ \mathbf{h}_i &= \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i \end{aligned} \tag{1}$$

where $\overrightarrow{\mathbf{h}}_i / \overleftarrow{\mathbf{h}}_i$ is the hidden state vector of i-th amino acid; $\overrightarrow{\mathbf{c}} / \overleftarrow{\mathbf{c}}$ are the memory cell states of i-th amino acid; $\overrightarrow{\mathbf{h}}_0, \overleftarrow{\mathbf{h}}_l$, $\overrightarrow{\mathbf{c}}_0$ and $\overleftarrow{\mathbf{c}}_l$ are initialized with random noise vectors; $\overrightarrow{\theta}^p$ and $\overleftarrow{\theta}^p$ are the learnable parameters of LSTM of forward and backward direction, respectively. With the embeddings of all the amino acids, we define the embedding for the peptide sequence as the concatenation of hidden vectors at two ends, that is, $\mathbf{h}^p = \overrightarrow{\mathbf{h}}_l \oplus \overleftarrow{\mathbf{h}}_0$.

To embed an MHC molecule into a continuous latent vector, we first flatten the encoding matrix E^m into a vector $\mathbf{m}$. Then, we learn the continuous latent embedding $\mathbf{h}^m$ with,

$$\mathbf{h}^m = W_1^m \text{ReLU}(W_2^m \mathbf{m}) \tag{2}$$

where W_i^m(i=1,2) are the learnable parameter matrices.

Action Prediction At time step t, we optimize the peptide sequence p_t by predicting the mutation of one amino acid with the latent embeddings $\mathbf{h}^{p_t}$ and $\mathbf{h}^m$. Specifically, we first select the amino acid a_i in p_t as the one to be replaced. We then predict which amino acid should be used to replace a_i. Inspired by [7], we add the current time step t as an additional feature into the prediction, so that the agent can optimize the peptides within the time limit T. We predict whether the amino acid a_i should be replaced, with the score of replacement calculated as below,

$$f^c(a_i) = (\mathbf{w}^c)^T (\text{ReLU}(W_1^c \mathbf{h}_i + W_2^c \mathbf{h}^m + w_3^c t)) \tag{3}$$

where $\mathbf{h}_i$ is the hidden latent vector of amino acid a_i from LSTM; w_3^c, $\mathbf{w}^c$ and W_i^c (i=1,2) are the learnable scalar, vector and matrices, respectively. The amino acid with the highest score will be selected to be replaced with another amino acid. After determining the amino acid a_i, we predict the type of the amino acid used to replace a_i as below,

$$f^d(a_i) = \text{softmax}(W_1^d \times \text{ReLU}(W_2^d \mathbf{h}_i + W_3^d \mathbf{h}^m + w_3^d t)) \tag{4}$$

where w_3^d and W_i^d (i=1,2,3) are the learnable scalar and matrices, respectively; softmax(.) converts a vector into probabilities over 20 amino acid types. Note that we will mask the original type of a_i and replace a_i with a different type of amino acid.

3.4 Optimization

We leverage PPO [9] to optimize the agent so that the peptides generated by the agent could have the maximum presentation score with a given MHC molecule. The objective function of PPO is defined as below,

$$\max J(\theta) = \tag{5}$$

4 Experimental Settings

4.1 Baseline Methods

To demonstrate the effectiveness of the proposed `PepPPO`, we develop multiple baselines as below and compare our method with them.

- `MCTS`: Monte Carlo Tree Search.

  Given an MHC molecule, `MCTS` generates peptide sequences by adding one amino acid step by step until reaching the maximum length of 15 amino acids or with the presentation scores greater than a threshold. All the generated peptides with no less than 8 amino acids will be evaluated by `MHCflurry2.0`.

- `BOVAE`: Bayesian Optimization with the Variational Autoencoder.

  In `BOVAE`, a VAE model is pre-trained to convert peptide sequences of variable lengths into continuous latent embeddings of a fixed size. Given an allele, the bayesian optimization algorithm with RBF kernel is then employed to optimize random latent embeddings with maximum t steps so that the peptide sequences decoded from the optimized latent embeddings are with the high presentation scores. At each step, `BOVAE` evaluates the presentation scores of the generated peptides with `MHCflurry2.0`, and stop the optimization and output the peptides if their presentation scores are greater than a threshold.

- `BPVAE`: Back Propagation with the Variational Autoencoder.

  In `BPVAE`, a student model with the variational autoencoder same as that in the `BOVAE` is employed to learn the presentation scores from `MHCflurry2.0`. After getting the trained student model, `BPVAE` can generate peptides by optimizing the latent embeddings of peptides through gradient ascent to maximize the predicted presentation scores in the student model. The decoded peptide sequences with t steps will be evaluated by `MHCflurry2.0`.

- `PWM`: Position weight matrix.

  To derive the position weight matrix for each allele, we used the data set for `MHCflurry2.0` curated by [6] to infer the position weight matrices. For each allele in the dataset, we counted the frequency of amino acids at each position in the positive peptides. The peptides are then generated by sampling from the frequency matrix.

- `Random`: Random peptide sequences of length from 8 to 15.

4.2 Dataset

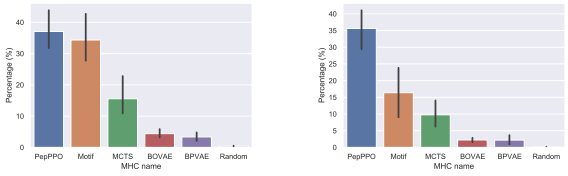
The dataset used to train the `MHCflurry2.0` and derive the `PWM` matrix is curated by [6], which combines both the IEDB dataset and the mass spectromy dataset. This combined dataset contains the binding affinity measurements or mass spectromy results of 672,613 peptide-MHC pairs with 371,954 unique peptides and 248 unique MHC class I molecules. The MHC molecules with the data only account for a very small portion of all the MHC molecules, since the number of MHC class I molecules in human (i.e., HLA) is 22,436 [4]. This demonstrates the importance and the difficulty of peptide generation for MHC molecules without the data, and also indicates that the baseline method `PWM` method could not perform well for these MHC molecules.

4.3 Experimental Setup

Parameter Setup We listed the hyper-parameters of the peptide mutation environment, the policy network and the RL agent in Table 1. We selected an optimal set of hyper-parameters including the discount factor, the entropy coefficient and the maximum steps by a grid search. We found little improvement when increasing the dimension or number of layers of policy network and value network. In the implementation of `PepPPO`, for each iteration, we ran 30 environments in parallel for 128 timesteps and collected trajectories with 3,840 timesteps in total. We set all the other hyperparameters for the RL agent as the default hyperparameters provided by the stable-baselines3 [8], as we found little performance improvement from tuning these parameters.

Table 1: Experimental Setup for Peptide Generation

description	value
maximum steps T	8
presentation score threshold σ	0.75
discount factor γ	0.9
hidden dimension of W_2^m	128
latent dimension of $\overrightarrow{\mathbf{h}}_i/\overleftarrow{\mathbf{h}}_i/\mathbf{h}^m$	48
hidden dimension of policy & value networks	40
hidden layers of policy & value networks	2
number of steps per iteration	3,840
entropy coefficient	0.001
batch size for policy update	64
number of epochs per iteration	10
clip range	0.2
learning rate	3e-4



(a) MHC Molecules with Data

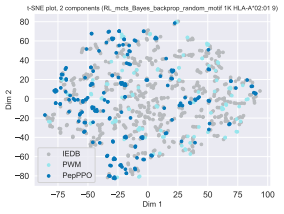
(b) MHC Molecules without Data

Figure 1: Performance Comparison on Average Percentage of Qualified Peptides over 1K Peptides for MHC Molecules with and without Data in the Dataset.

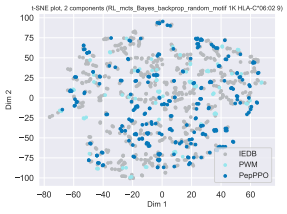
5 Experimental Results

5.1 Overall Comparison

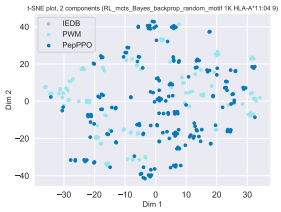
We sampled a set of MHC Class I molecules with and without the data, and compared the performance of all the methods on generating peptides bound to the MHC molecules. Each method is used to generate 1K peptides for each MHC molecule. Figure 1a and Figure 1b present the performance comparison on the generation of peptides bound to MHC molecules with and without data, respectively. As shown in the figures, `PepPPO` achieves the highest percentages of qualified peptides (i.e., with presentation scores greater than 0.75) among all the methods. For the MHC molecule with the data, `PepPPO` outperforms the best baseline `PWM` by about 3%. The good performance of `PWM` on these MHC molecules could be due to that the frequency matrices derived from



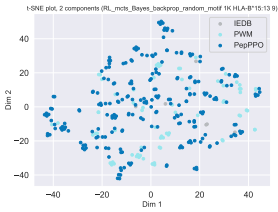
(a) HLA-A*02:01 with data



(b) HLA-C*06:02 with data



(c) HLA-A*11:04 without data



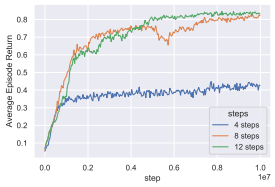
(d) HLA-B*15:13 without data

Figure 2: t-SNE plot of the generated peptides for MHC molecule HLA-A0201 and HLA-C0602

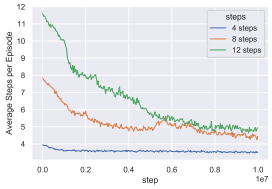
enough qualified peptides (i.e., 1.6K - 26K in the dataset) can accurately describe the distribution of qualified peptides of these MHC molecules. The better performance of PepPPO over PWM could demonstrate that PepPPO can learn the additional patterns of qualified peptides. For MHC molecules without the data, PepPPO significantly outperforms the best baseline PWM by about 20%. When the MHC molecules don't have enough qualified peptides in the dataset and are not similar with any other MHC molecules with the sufficient peptides, PWM cannot find the position weight matrices from the dataset that could accurately describe the distribution of qualified peptides of these MHC molecules, leading to the worse performance on generating qualified peptides. Instead, PepPPO learns a common policy network to generate peptides bound to any random MHC molecules, and thus, achieves comparable performance on MHC molecules with the data (i.e., 36%). Generating qualified peptides for MHC molecules without the data is more important than that for MHC molecules with thousands of peptides known to be qualified. Therefore, PepPPO could be used to generate quantified peptides of MHC molecules.

5.2 Case Study

In Figure 2, we plotted four t-SNE plots of the generated qualified peptides in the above experiments for two MHC molecules HLA-A*02:01 and HLA-C*0602 with enough qualified peptides in the dataset and two MHC molecules HLA-A*11:04 and HLA-B*15:13 without. From Figure 2a and Figure 2b, we observed that both PWM and PepPPO can cover the space of qualified peptides populated by the existing qualified peptides (i.e., binding affinity < 500) in the dataset. This demonstrates that the generated qualified peptides of PepPPO have similar distributions with those in the dataset.



(a) Average Final Presentation Scores of RL agents



(b) Average Episode Length of RL agents

Figure 3: Comparison of RL Agents with Different Maximum Steps

6 Ablation Studies

6.1 Threshold

6.2 Maximum Length of Steps

Figure 3 presents the performance comparison of RL agents with the maximum 4 steps, 8 steps and 12 steps on the average presentation scores (Figure 3a) and the average length for each episode (Figure 3b). From Figure 3, we can conclude

7 Discussion

7.1 Peptide Mutation Policy

Two examples, one with data and another without data. Plot the distributions of the predicted actions.

8 Conclusion

References

[1] C. Angermüller, D. Dohan, D. Belanger, R. Deshpande, K. Murphy, and L. Colwell. Model-based reinforcement learning for biological sequence design. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[2] Z. Chen, M. R. Min, and X. Ning. Ranking-based convolutional neural network models for peptide-MHC class i binding prediction. *Frontiers in Molecular Biosciences*, 8, may 2021.

[3] V. Jurtz, S. Paul, M. Andreatta, P. Marcatili, B. Peters, and M. Nielsen. NetMHCpan-4.0: Improved peptide–MHC class i interaction predictions integrating eluted ligand and peptide binding affinity data. *The Journal of Immunology*, 199(9):3360–3368, oct 2017.

[4] S. G. E. Marsh, E. D. Albert, W. F. Bodmer, R. E. Bontrop, B. Dupont, H. A. Erlich, M. Fernández-Viña, D. E. Geraghty, R. Holdsworth, C. K. Hurley, M. Lau, K. W. Lee, B. Mach, M. Maiers, W. R. Mayr, C. R. Müller, P. Parham, E. W. Petersdorf, T. Sasazuki, J. L. Strominger, A. Svejgaard, P. I. Terasaki, J. M. Tiercy, and J. Trowsdale. Nomenclature for factors of the HLA system, 2010. *Tissue Antigens*, 75(4):291–455, apr 2010.

[5] M. Nielsen, C. Lundegaard, T. Blicher, K. Lamberth, M. Harndahl, S. Justesen, G. Røder, B. Peters, A. Sette, O. Lund, and S. Buus. NetMHCpan, a method for quantitative predictions of peptide binding to any HLA-a and -b locus protein of known sequence. *PLoS ONE*, 2(8):e796, aug 2007.

[6] T. J. O'Donnell, A. Rubinsteyn, and U. Laserson. Mhcflurry 2.0: Improved pan-allele prediction of mhc class i-presented peptides by incorporating antigen processing. *Cell Systems*, 11(1):42–48.e7, 2020.

[7] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev. Time limits in reinforcement learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4045–4054. PMLR, 10–15 Jul 2018.

[8] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[10] M. J. Skwark, N. L. Carranza, T. Pierrot, J. Phillips, S. Said, A. Laterre, A. Kerkeni, U. Sahin, and K. Beguir. Designing a prospective COVID-19 therapeutic with reinforcement learning. *CoRR*, abs/2012.01736, 2020.