Model Architecture

1. Environment

State : allele sequence; peptide sequence

Action :

1. stop or not: determine whether or not to stop the edit of sequence. (stop edits when stop == 1)
2. position: determine the position of edits (**replace one amino acid with another one**).
3. amino acid: determine the type of amino acid at the predicted position.

2. Policy Network $\pi(s)$

2.1 Representation Learning

Amino acid embedding: Bi-LSTM

For each amino acid, we will represent it with the forward and backward embeddings.

$$h_i^* = \overrightarrow{f}(h_{i-1}^*, x_i); b_i = \overleftarrow{f}(b_{i+1}, x_i)$$

$$h_i = Concat(h_i^*, b_i)$$

For each allele sequence, we can select one of model architecture from CNN/RNN/MLP/Transformer to embed the allele sequence as a. (use MLP and CNN for parameter tuning.)

2.2 Action Prediction

Stop prediction

Predict whether stopping the edit of sequence or not using the embeddings of peptide sequences and alleles

$Softmax(m_1(h_n, a))$

Amino acid position selection

Given a peptide sequence of length n, we can learn its embedding $p_t = (h_0, \ldots, h_{n-1}) \in R^{n \times k}$. (**add** position encoding here)

$f_i = Softmax(m_2(h_i, a, pos_i))$

Amino acid type prediction

predict the type of added amino acid as below,

$f_2 = Softmax(m_3(h_i, a))$

m_3 will output an vector of length 20.

The probabilities of illegal actions should be masked.

3. Model training

Use PPO to optimize the model.

Sampling Method (used to sample the initial state):

All peptides are sampled from intermediate or negative binding peptides in IEDB dataset.

Some basic settings:

- feature type: blosum + onehot + deep
- latent dimension: (40, 20)
- value network and policy network: (80, 40)
- architecture for allele sequence: (CNN, FC)
- entropy coefficients: (0.0, 0.001, 0.01, 0.1)
- Hidden dimension for allele: (320)
- latent dimension for allele: (80)
- latent dimension for peptide (Bi-LSTM): (40)

Value Network $V(s)$

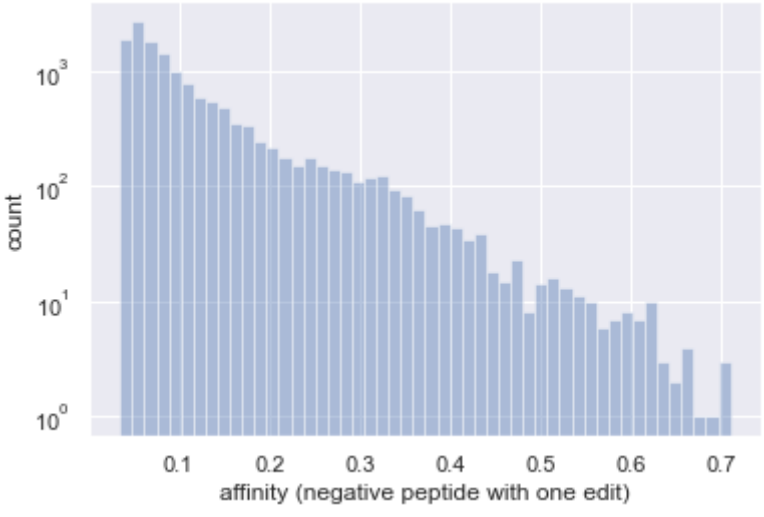
MLPs: Given a peptide sequence, predict the estimation of its future reward.

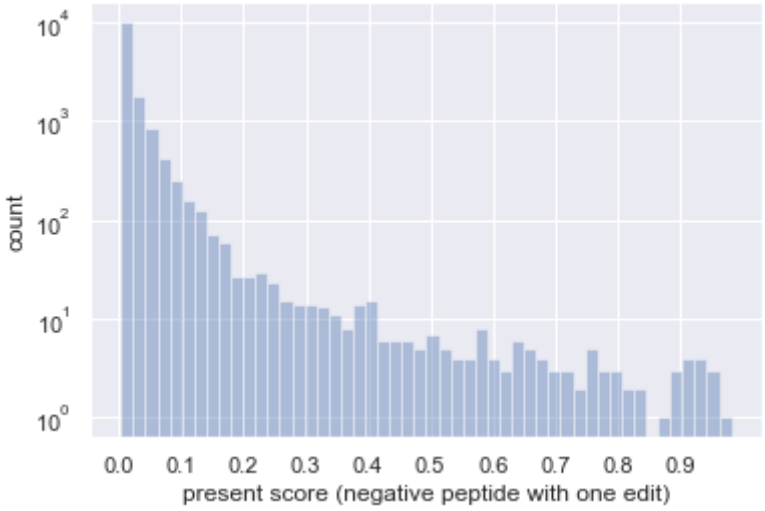
4. Reward Design

Analysis about the presentation scores predicted by MHCflurry:

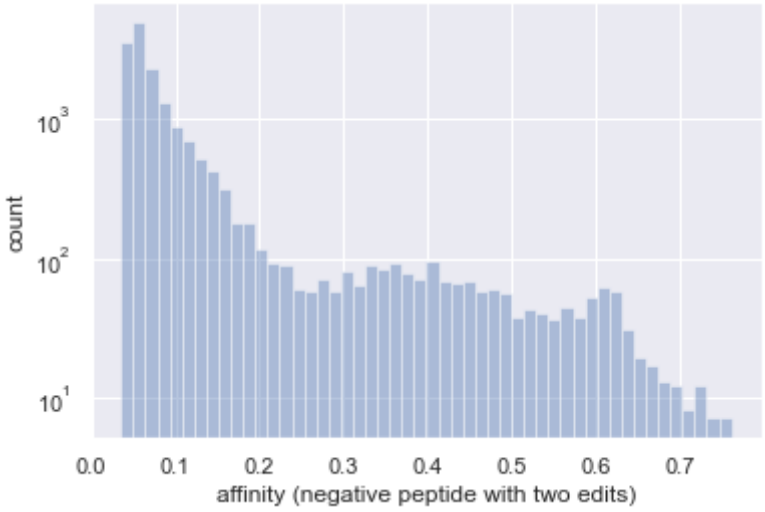
4.1 Negative Sample

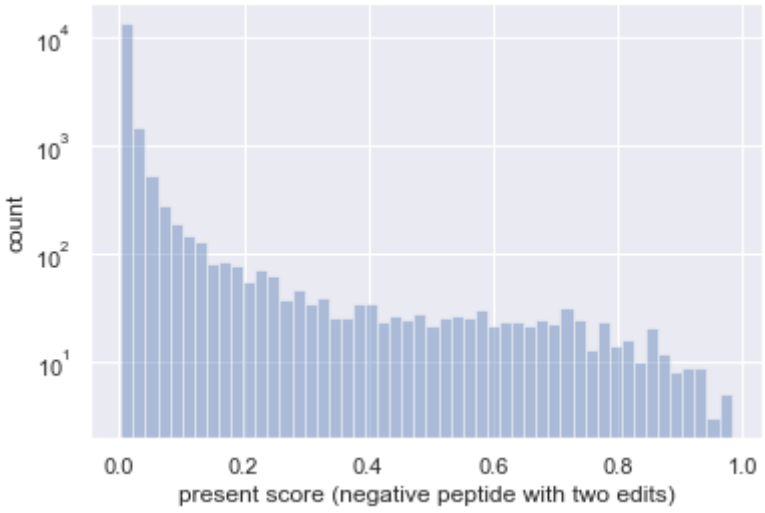
We sampled 100 peptides with binding affinity > 5000, and kept only samples with predicted presentation scores < 0.1. Then, we edited these peptides by replacing one amino acid and calculated the predicted affinities and presentation scores of new sequences.





We sampled 100 peptides from the edited sequences, and edited these peptides again. Then, We calculated the predicted affinities and presentation scores of new sequences.

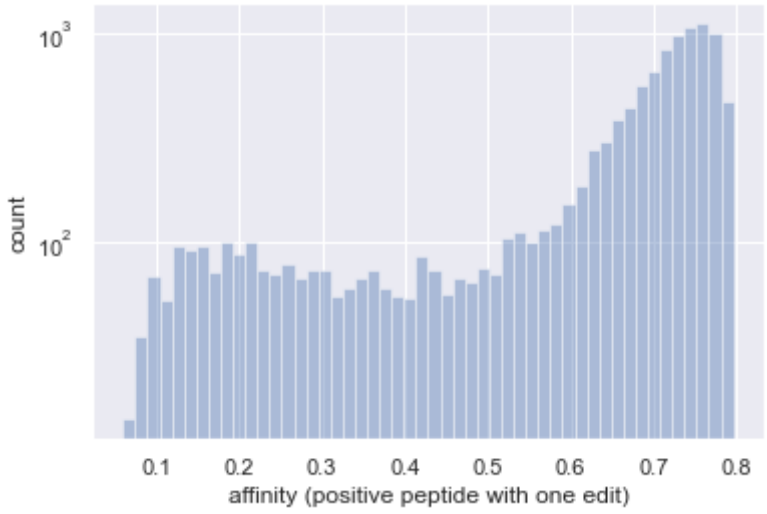


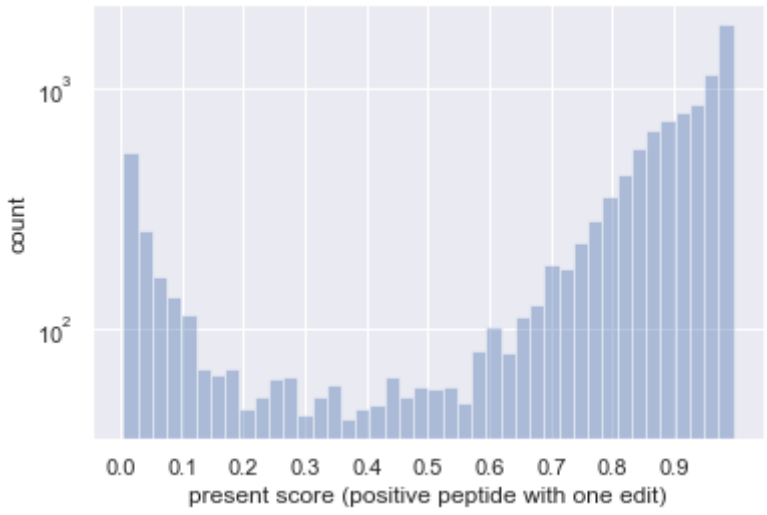


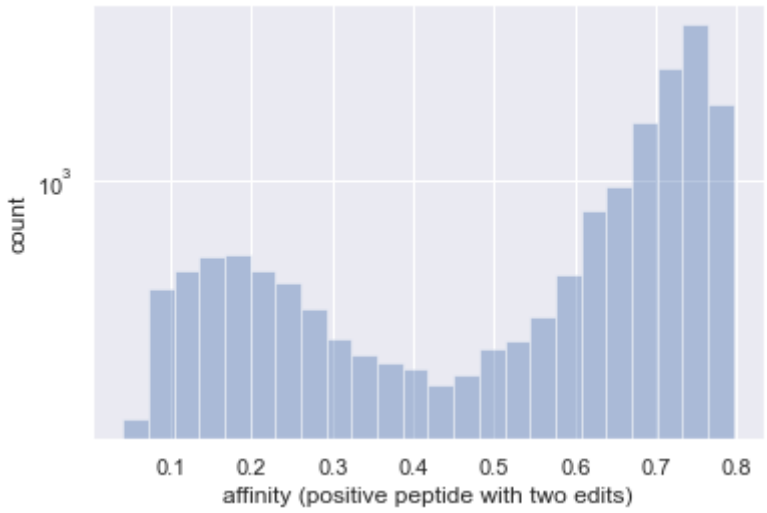
Conclusion: If we start from negative peptides, it is very likely that we can hardly receive any rewards (always 0 - 0.05).

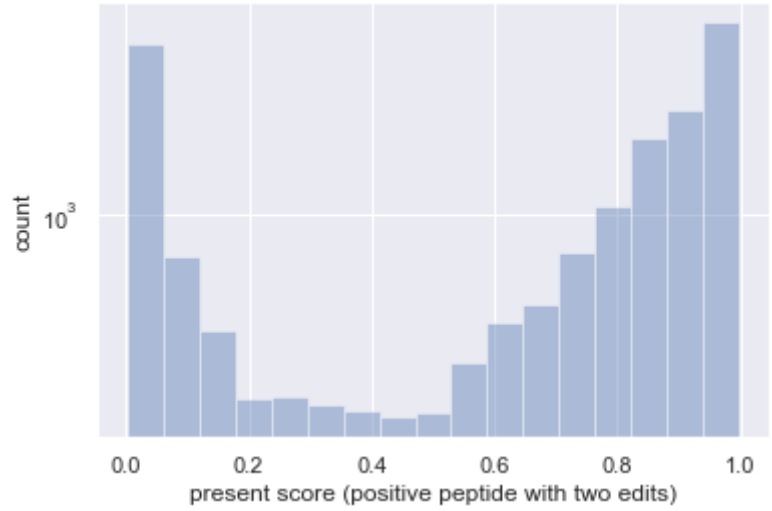
4.2 Positive Sample

We sampled 100 peptides of length 9 binding to the allele HLA-A*02:01 with binding affinity < 100, and kept only samples with predicted presentation scores > 0.8. Then, I edited these peptides by replacing one amino acid and calculated the predicted affinities and presentation scores of new sequences.



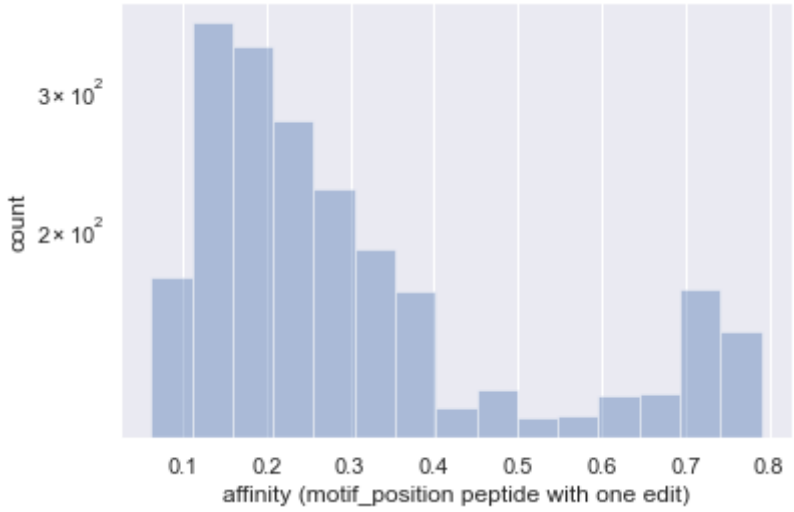


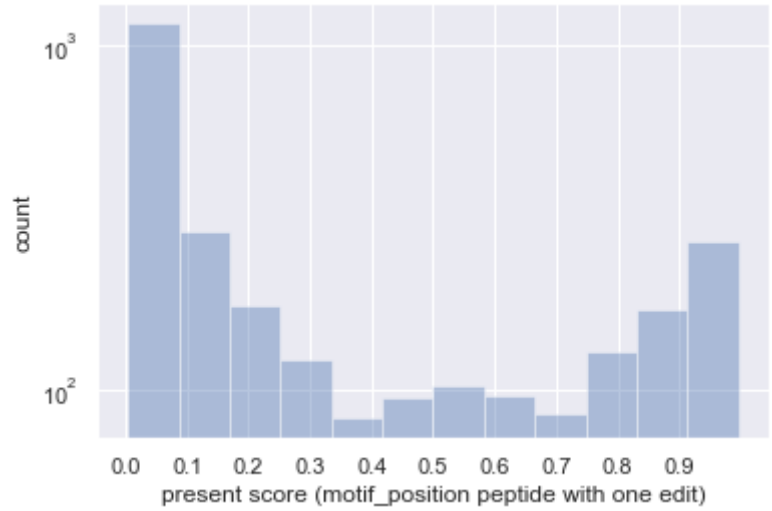


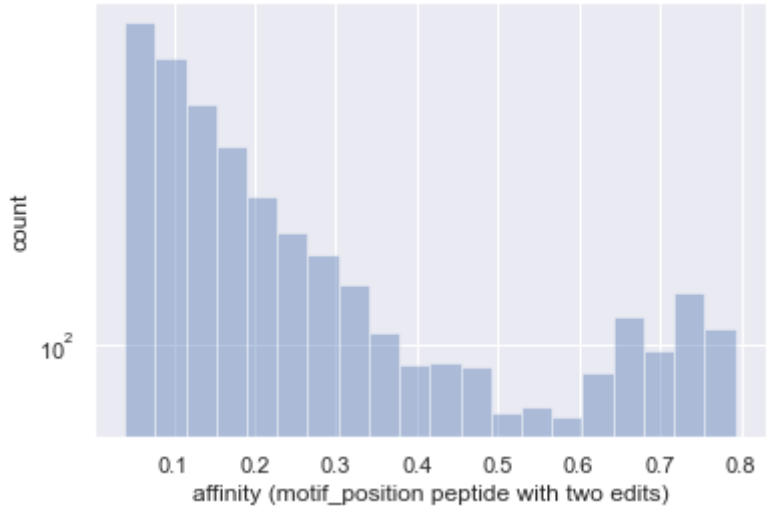


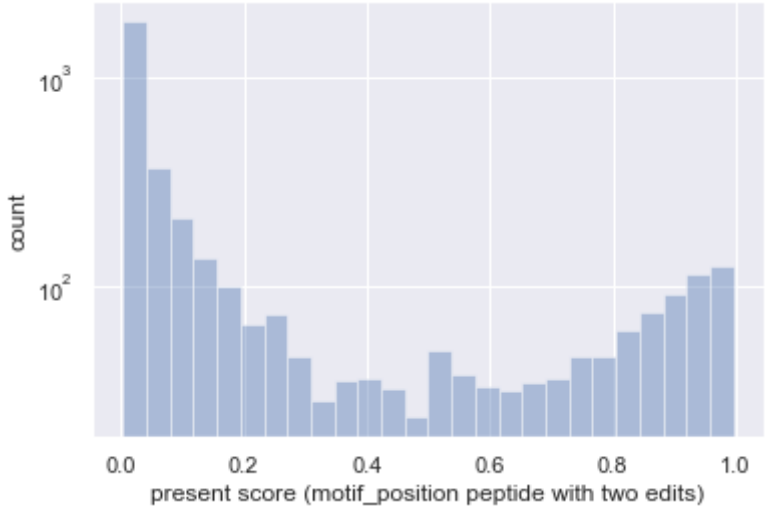
Motif Position

If we only edit the amino acids at the motif positions in peptides,



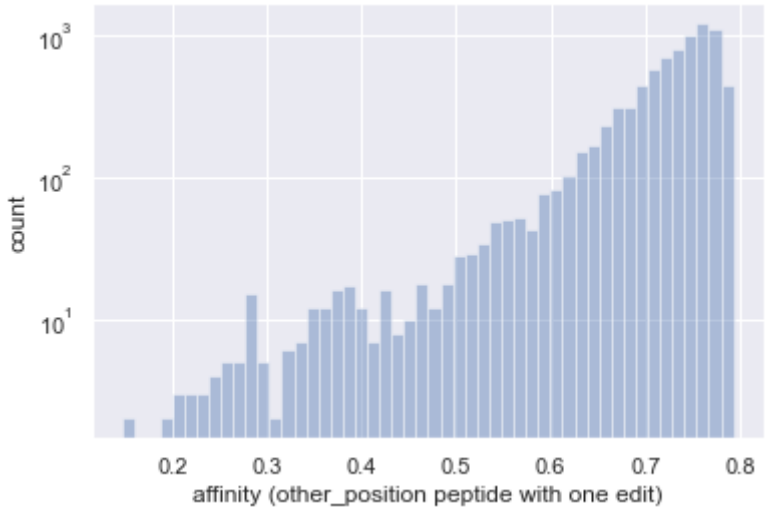


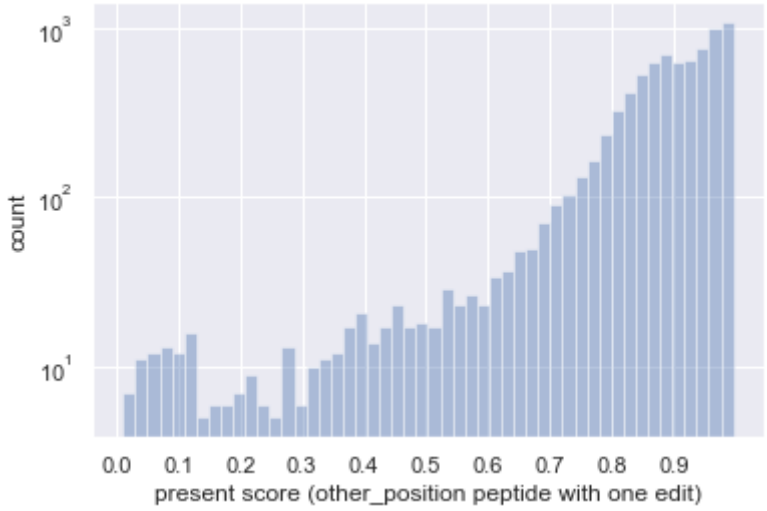


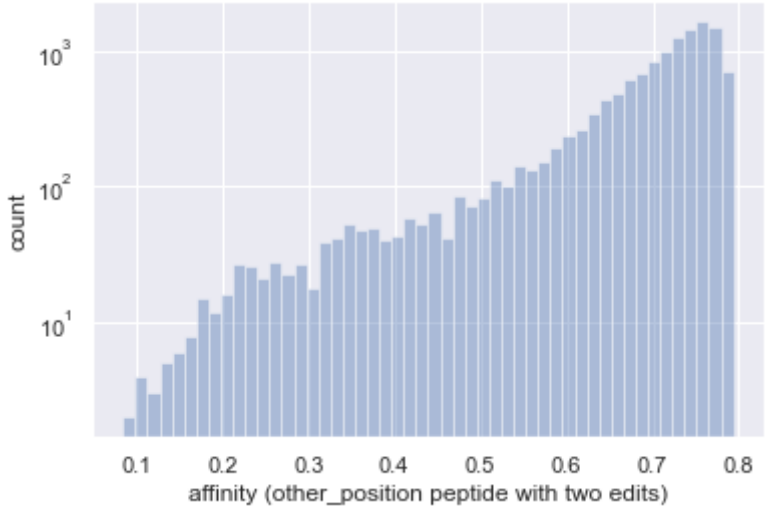


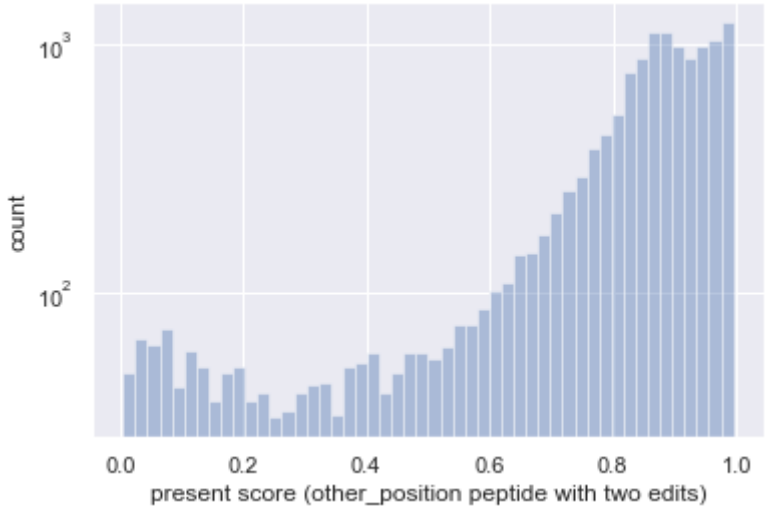
Other position

If we edit the amino acids at the non-motif positions in peptides,









4.3 Reward Design

- Maze game and Mountain Car

  For each step that the agent does not reach the goal, the environment returns a negative reward (penalty) to avoid long path.

Note: The end of edit process cannot be determined or stopped by policy network; otherwise, the policy network can always tend to stop the edit process as soon as possible. That means, we need to set a goal so that the environment can stop the edit process by either reaching the goal or reaching the maximum steps.

[Our reward design]:

Intermediate reward: -0.3

Final reward: 10 * presentation score

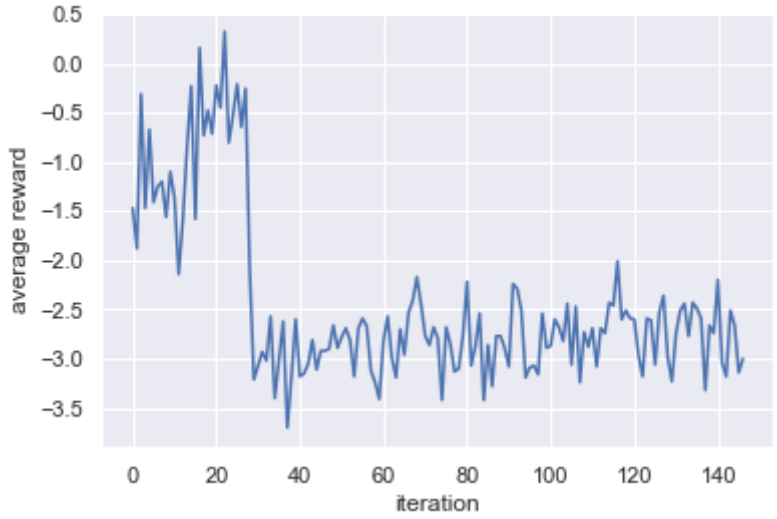
Maximum steps: 15

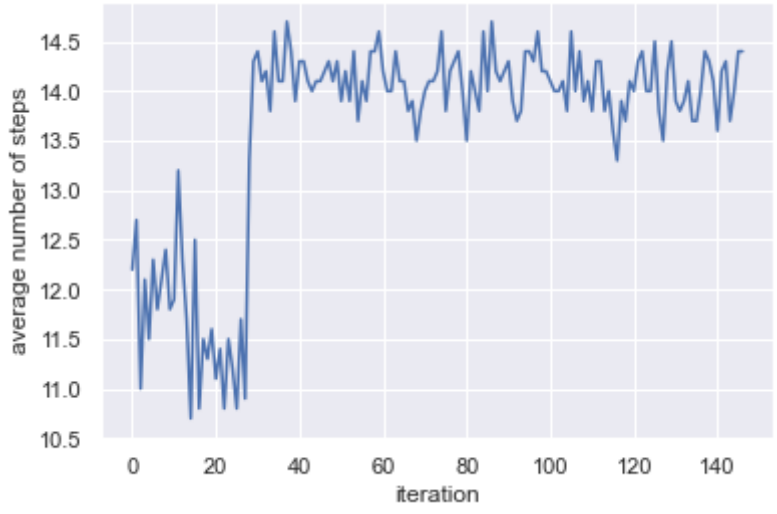
Goal: 0.6

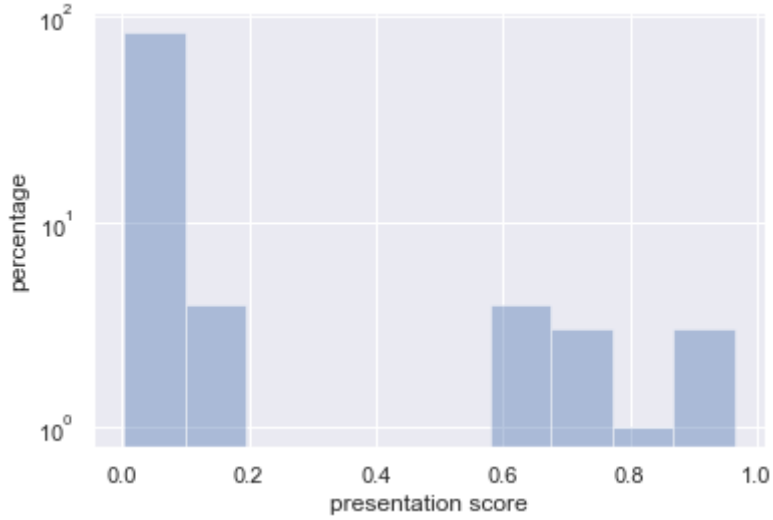
Initial good samples: peptides with presentation score from 0.2 to 0.5, with the assumption that these peptides are easier to be optimized and thus provide more signals than random peptides.

Results:

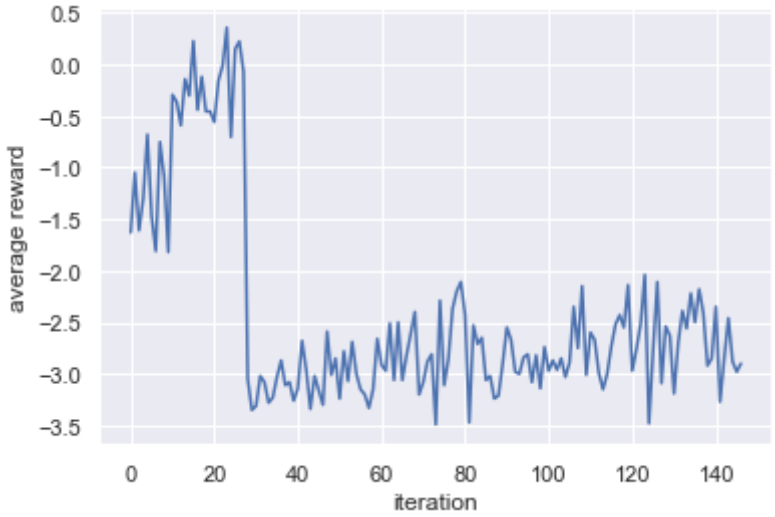
With CNN:

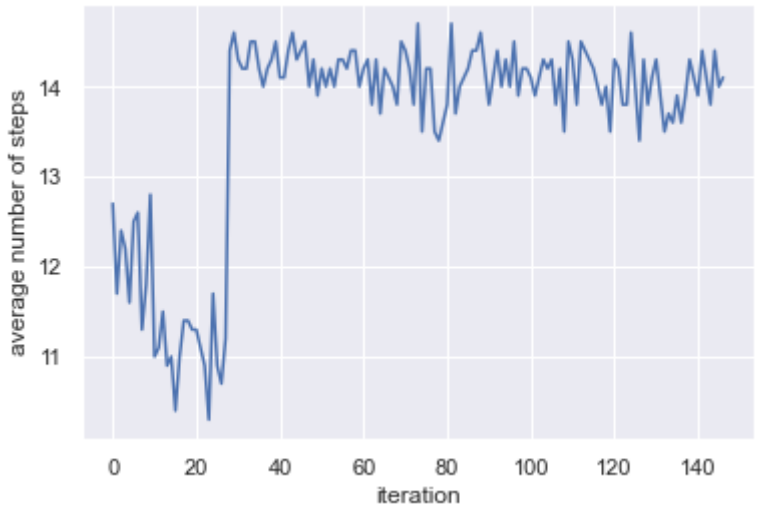


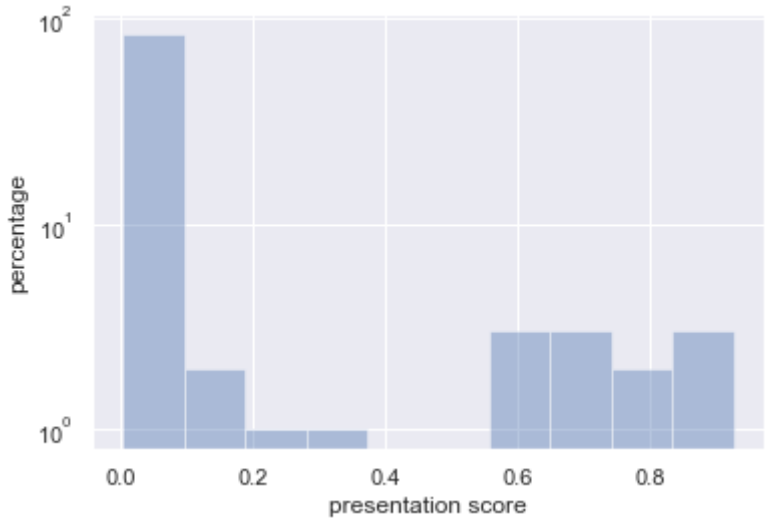



10^2
10^1
10^0
percentage
0.0
0.2
0.4
0.6
0.8
1.0
presentation score

With FC:


0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
average reward
0
20
40
60
80
100
120
140
iteration


14
13
12
11
average number of steps
0
20
40
60
80
100
120
140
iteration



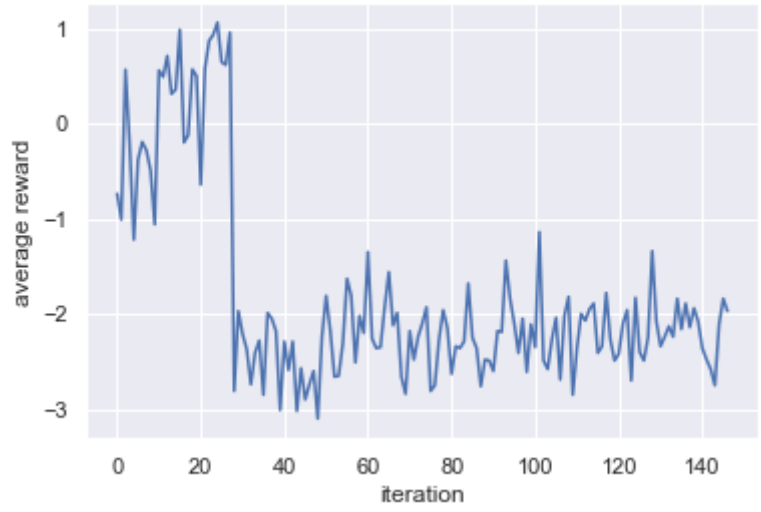
[Our reward design]:

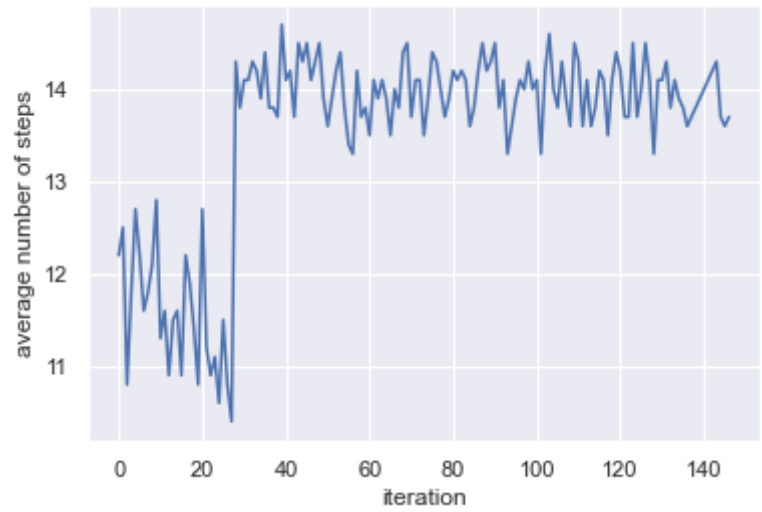
Intermediate reward: -1 + presentation score

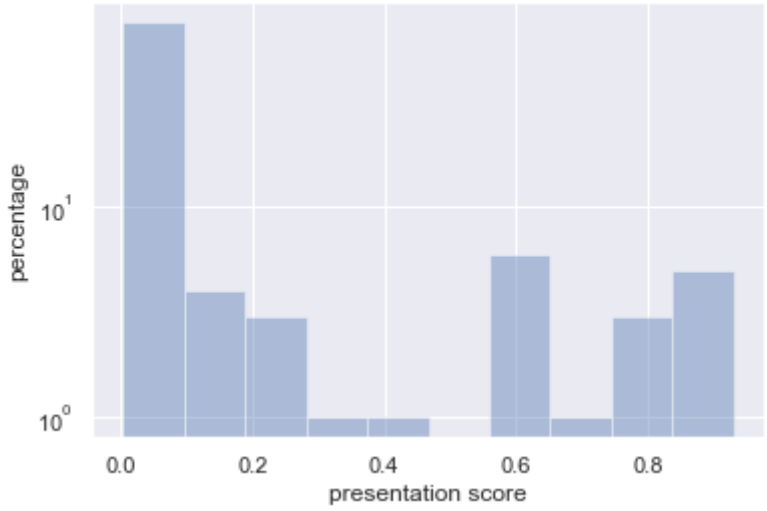
Final reward: 10 * presentation score

Maximum steps: 15

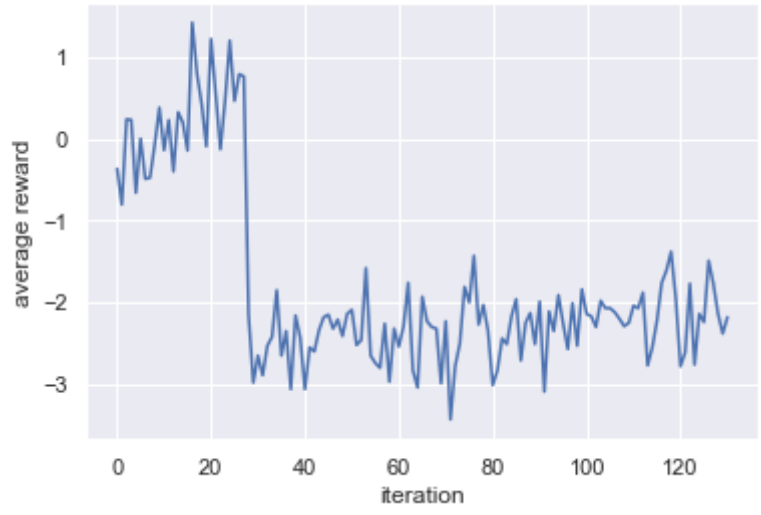
With CNN:

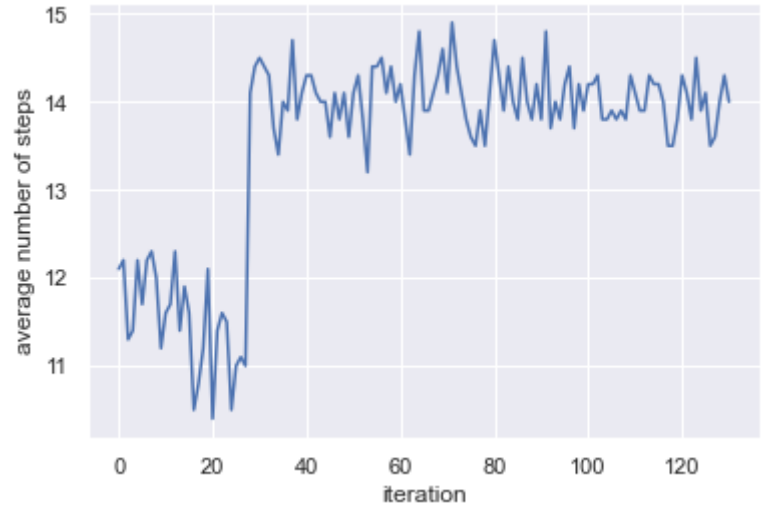


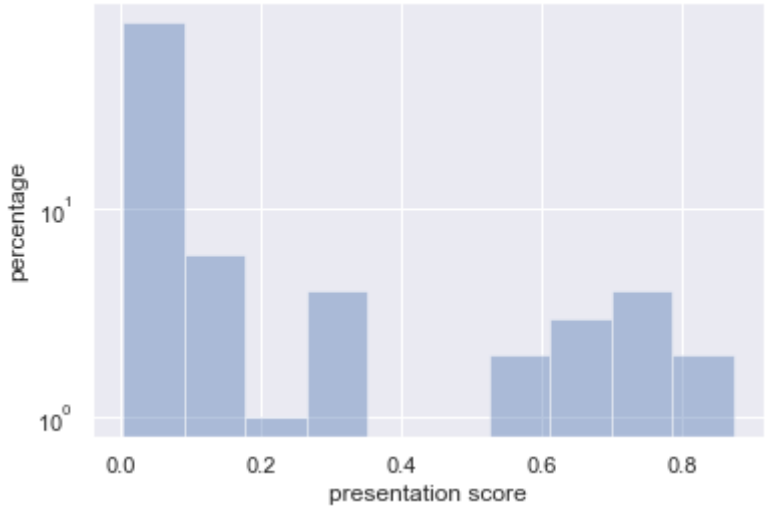



percentage
10^1
10^0
0.0
0.2
0.4
0.6
0.8
presentation score

With FC:


average reward
1
0
−1
−2
−3
0
20
40
60
80
100
120
iteration


average number of steps
15
14
13
12
11
0
20
40
60
80
100
120
iteration



- Molecule modification

  For each step, the environment returns a property score as a reward.

  The environment defines the maximum number of steps T

  The agent can choose "no modification" action to keep the molecules unchanged until the step T.

  At step t, the reward will be discounted by β^{T-t} and $\beta < 1$, so that the final reward will have the largest contribution to the cumulative reward.

- Other directions
  - choose the maximum reward

    finite horizon undiscounted return: $R(\tau) = \sum_{t=0}^{T} r_t$

    infinite horizon discounted return: $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$

    The corresponding Bellman equation for $Q^\pi(s,a)$ with the expected return defined as $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$ is

    $$Q^\pi(s_t, a_t) = \mathbb{E}_{\substack{s_{t+1} \sim P(\cdot|s_t, a_t) \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} \left[r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) \right]$$

    optimize maximum reward achieved in a trajectory,

    $R(\tau) = \max_{t \geq 0} \gamma^t r_t$

    $\gamma < 1$ allows the agent to prefer rewards sooner rather than later.

    The action-value function for a stationary policy π, denoted $Q^\pi_{\max} : \mathcal{S} \times A \to \mathbb{R}$, represents the expected maximum of discounted rewards along the trajectories induced by the policy in the MDP:

    $$Q^\pi_{\max}(s_t, a_t) \triangleq \mathbb{E}\left[\max_{t' \geq t} \left(\gamma^{t'-t} r(s_{t'}, a_{t'}) \right) \mid (s_t, a_t), \pi \right], \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A} \quad (2)$$

- New settings (discussion on 3/3)

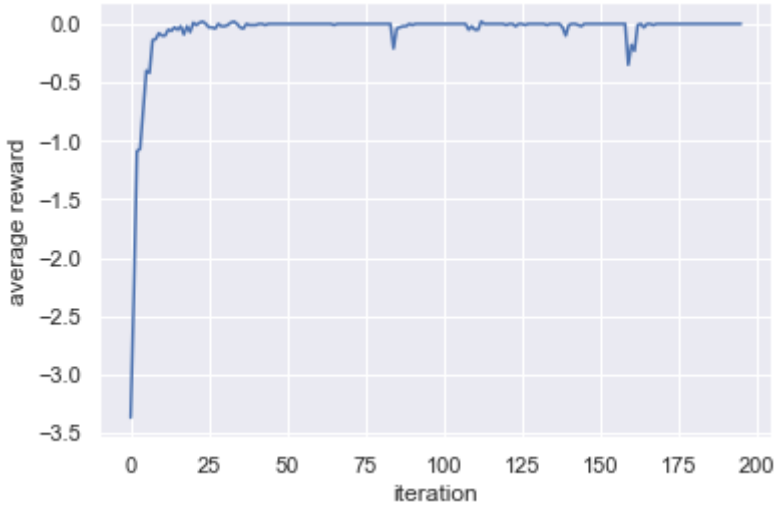
  action: 3 different actions (no modification/ modification; position; type)

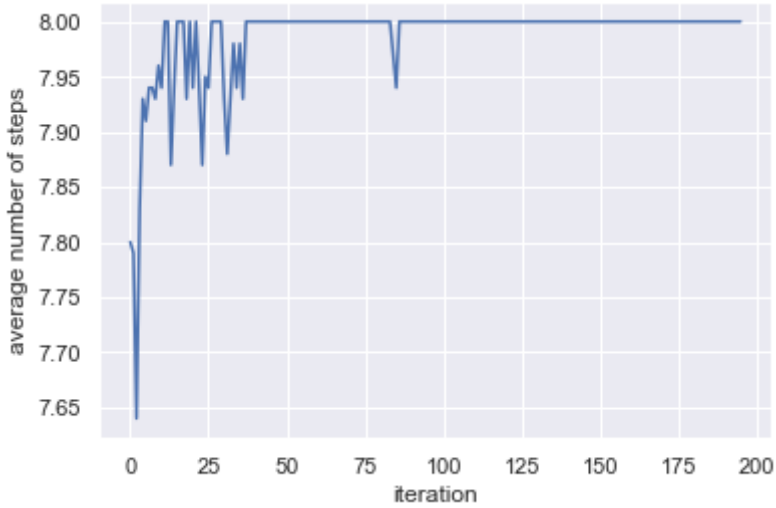
  intermediate reward r_t: +1 (increase larger than δ); -1 (increase smaller than δ); 0 (no modification); ($\delta = 0.1$)

  stop: reach the goal / maximum steps

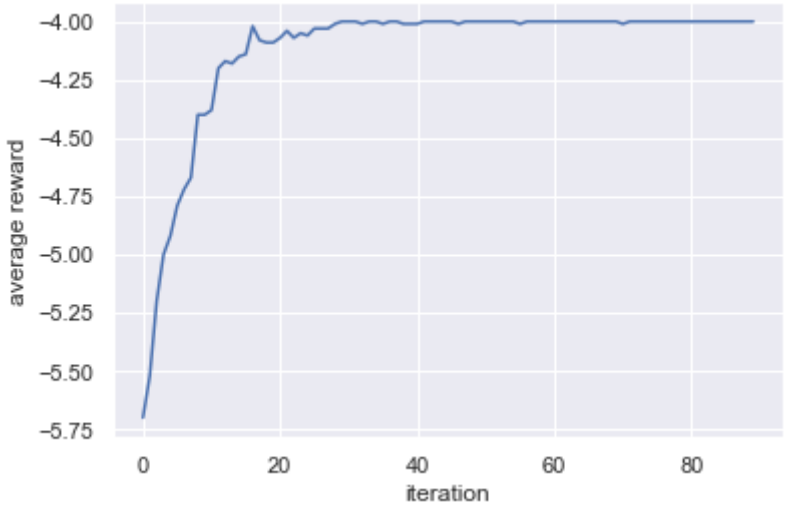
  use this setting for pre-training, and then use the setting with only final rewards
  - Results with (no final reward + no penalty for "no modification" action)

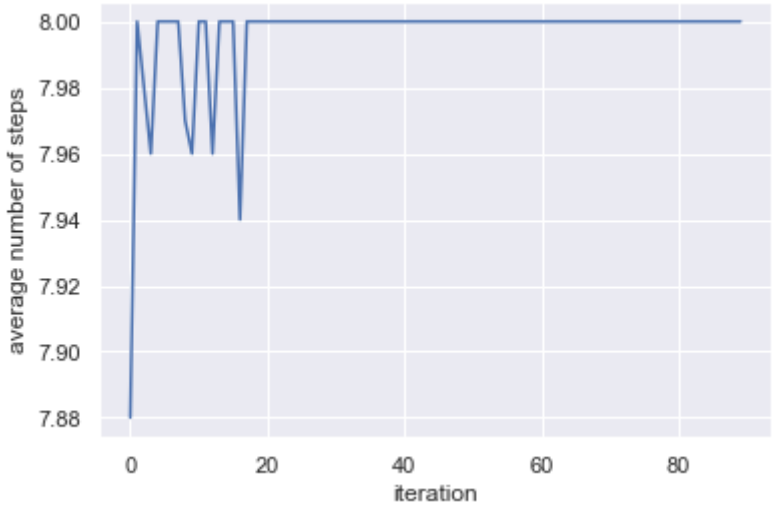
    The policy stuck into the "no modification" actions....



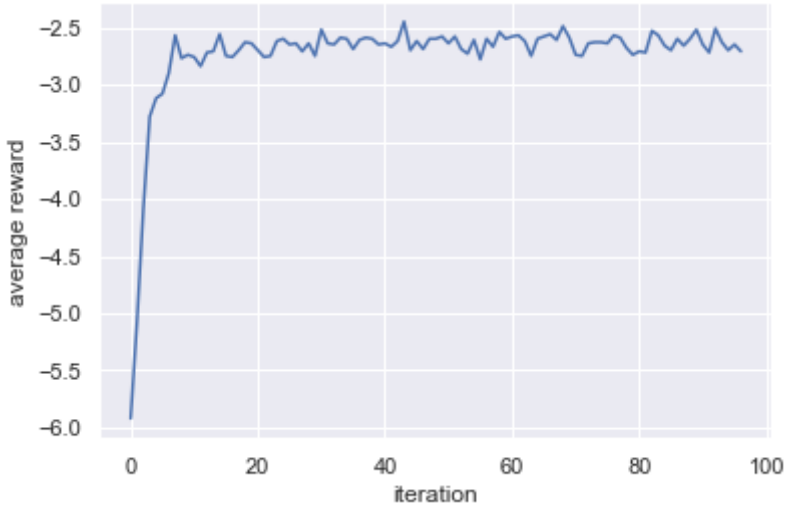


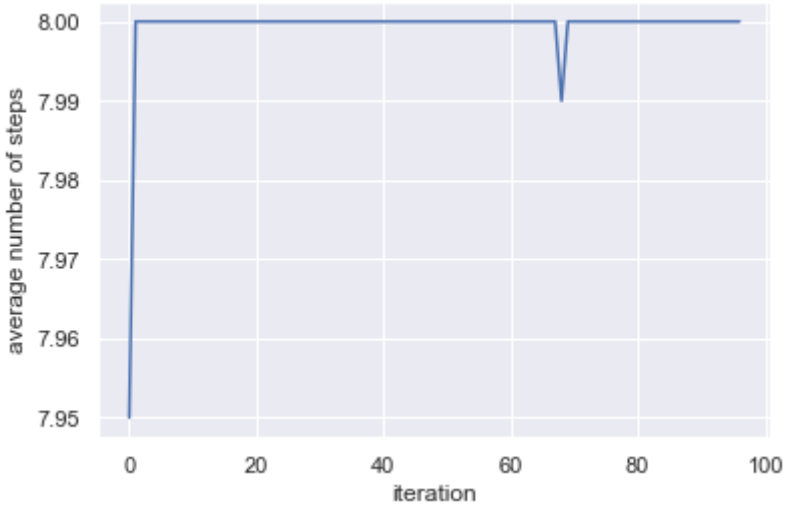
- Results with (no final reward + penalty for "no modification action") (penalty = -0.5)



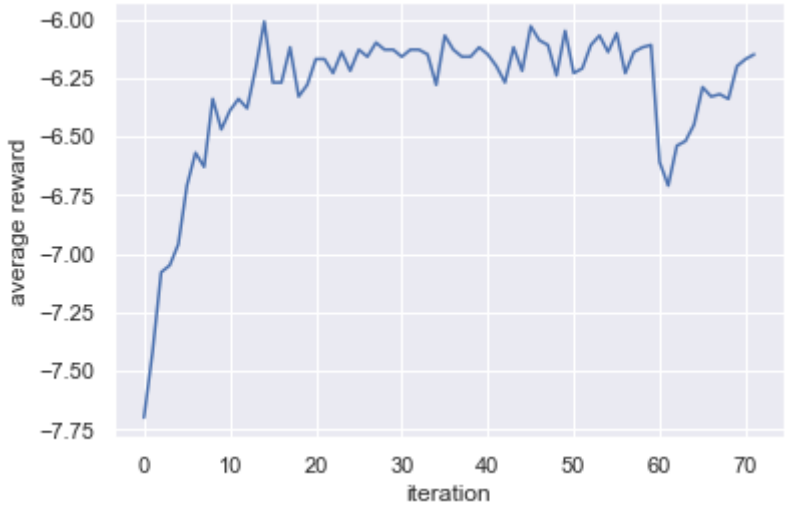


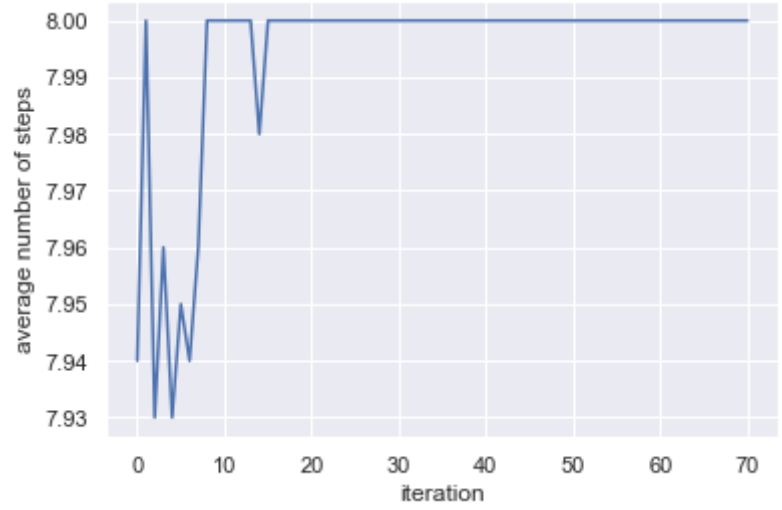
- Results with (final reward score * 10 + no penalty for "no modification action")



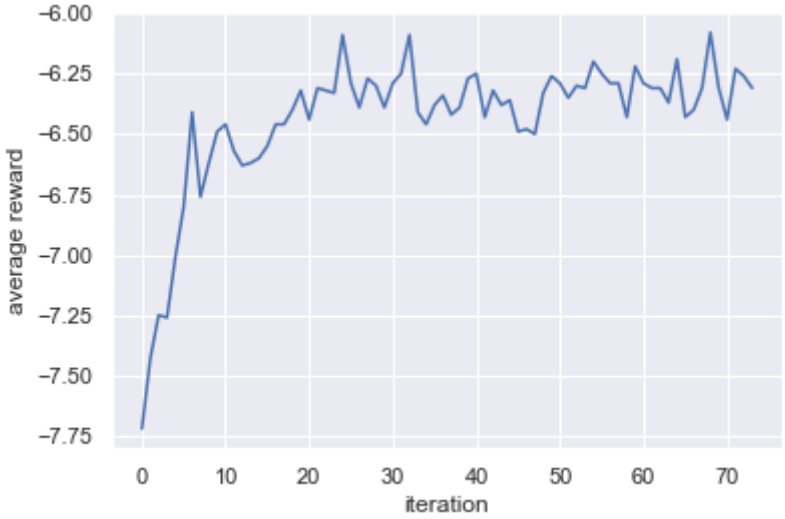


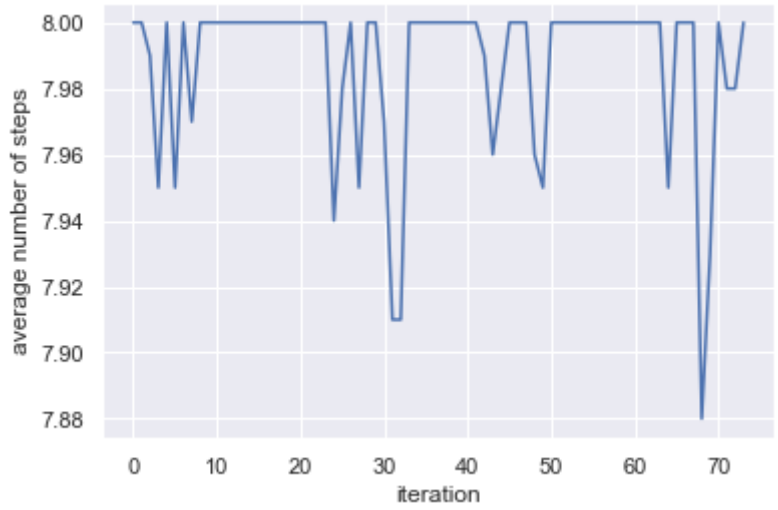
- Results with (final reward score * 10 + penalty for "no modification action") (penalty = -0.5)





with large coefficients for entropy loss to encourage exploration:





- Pretrain policy network. (generate sequences for pretraining?)
- Edit motif position.
- remove "no modification" action from the action space and remove final reward.
- schedule exploration. use a large exploration ratio at the beginning.

Update (4-14)

Add prior distribution into the sampling.

Given an allele, find positive peptides at different length from the IEDB dataset.

If the number of positive peptides at length l for allele a is greater than 10, then we derive the distribution of amino acids at each position; otherwise, we find the most similar allele (calculated from BLOSUM62 matrix) with positive peptides greater than 10 and directly use the distribution of that most similar allele.

After deriving the distribution of amino acids at each position, we can use the entropy of amino acid distribution to demonstrate whether or not those positive peptides have preference on some specific amino acids at the position. In other words, we can use the entropy to identify the motif position.

Therefore, with such a distribution, we can get the motif-position probability distribution $p_p(pos|a)$ using the entropy, and get the amino acid distribution $p_m(amino|a,pos)$ at a specific position.

At the first 50k steps, we directly combine the derived prior distribution with the calculated distribution from policy network.

$pos \sim \text{Categorical}(p_p(pos|a,l) + p_\theta(pos|p,a))$

$amino \sim \text{Categorical}(p_m(amino|a,l,pos) + p_\theta(amino|p,a,pos))$

(use these two distributions separately)

Reward Design:

action: 3 different actions (no modification/ modification; position; type)

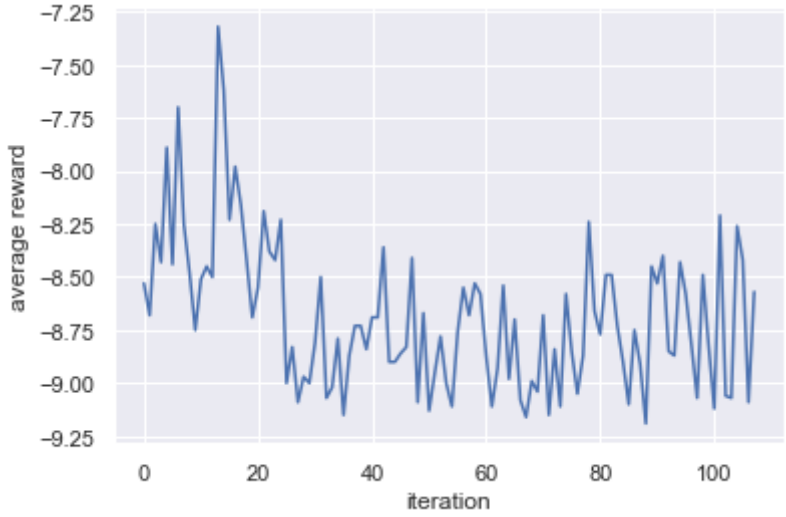
intermediate reward r_t: +1 (increase larger than δ); -1 (increase smaller than δ); -0.9 (no modification); ($\delta = 0.05$)

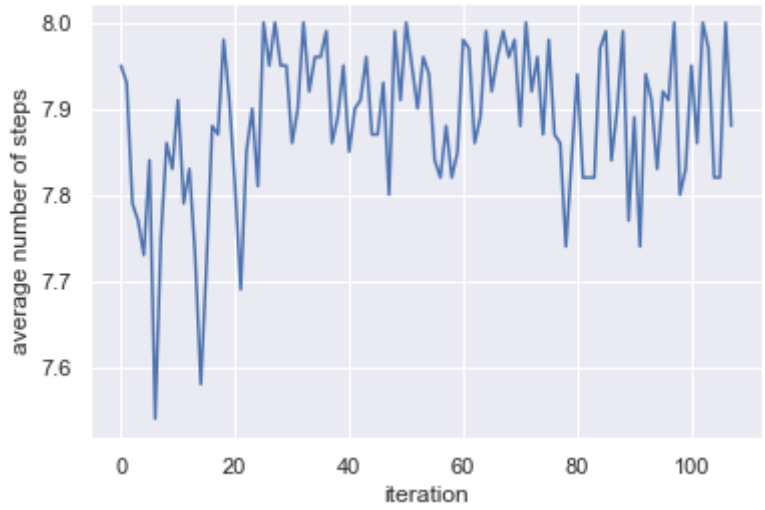
Final reward: present score * 10

Max steps: 8

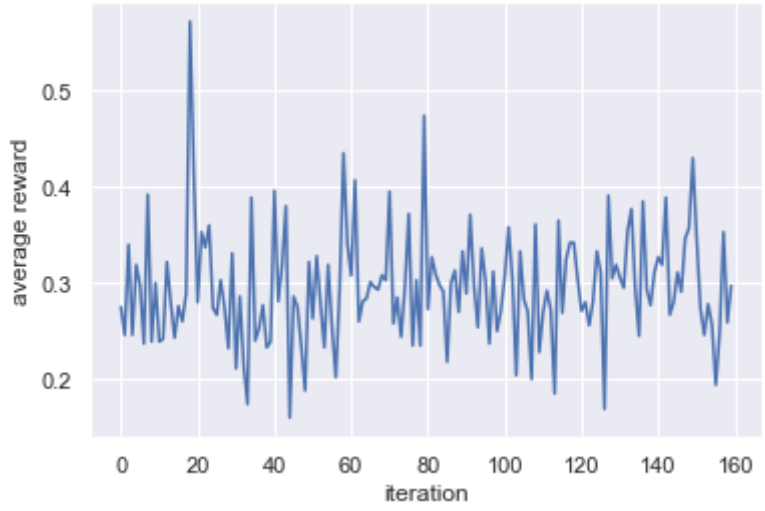
Goal: 0.6

50K step ~ 25 iteration with prior distribution.





Another setting from molecule modification (this setting doesn't perform very well....): γ^{T-i}



Conclusion:

The usage of prior distribution seems to be incorrect.

- either use the prior distribution or use the distribution from policy network to sample trajectories.
- use the prior distribution for more iterations.

Update (4-21)

Objective:

Free generation without given alleles.

Given the Covid peptides, we find the mutated peptides for the given alleles?

Search space can be large even with the hard constraint (amino acid difference) or the soft constraint (Blosum62 matrix distance).

Plan:

1. Train the model for more iterations.
2. Update the prior distribution during training.

- The usage of prior distribution:

  At each step, we sample the actions from the prior distribution with probability β, and from the distribution predicted by policy networks with probability $(1-\beta)$.

  We set the initial β = 0.5 and decrease the value of β by 0.05 every 20k steps.

  Each iteration has 2,048 steps. $\beta = 0$ at iteration 100.

- A potential problem with the prior distributions is that:

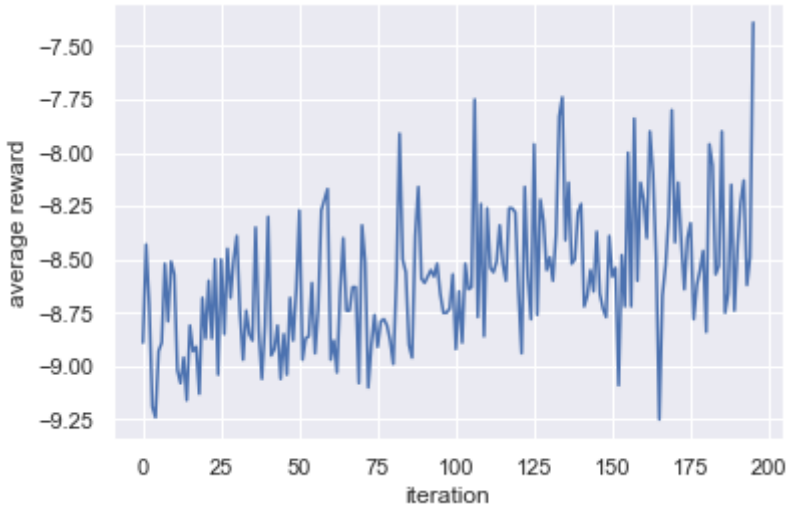
  The derived prior distributions from the dataset can be very sparse.

  > For allele "BoLA-2*12:01", the positive peptide sequences at length 11 in IEDB dataset are very similar:

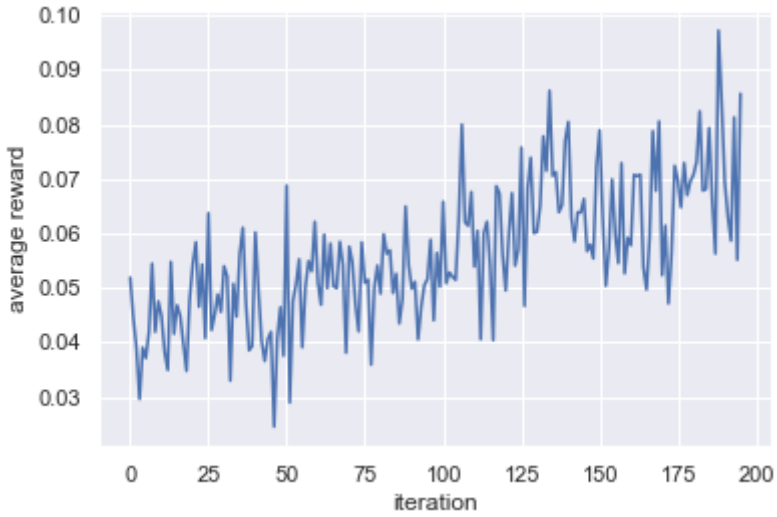
> 'KAAHGMGKVGK', 'KASHGMGKVGK', 'KSAHGMGKVGK', 'KSKHGMGKVGK', 'KSSAGMGKVGK', 'KSSHAMGKVGK', 'KSSHGAGKVGK', 'KSSHGMAKVGK', 'KSSHGMGAVGK', 'KSSHGMGEVGK', 'KSSHGMGKAGK', 'KSSHGMGKVAK'

- Setting 1 (+1, -1, -0.9)

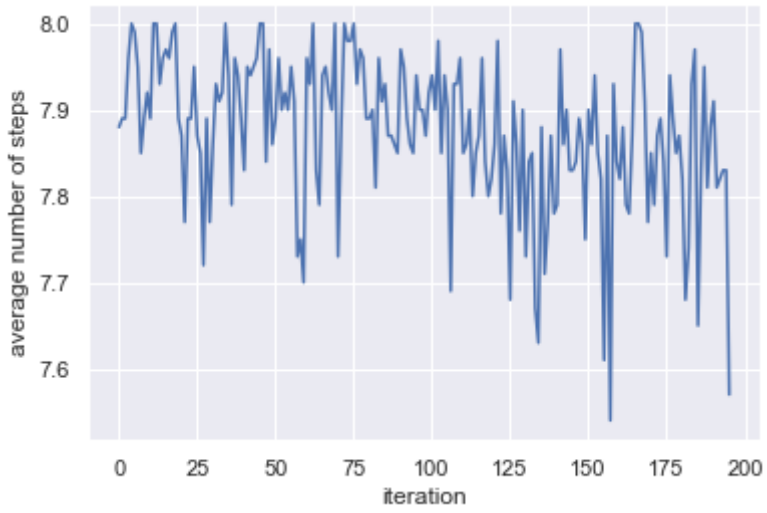
Cumulative reward (return):



Final reward /10 (presentation score):

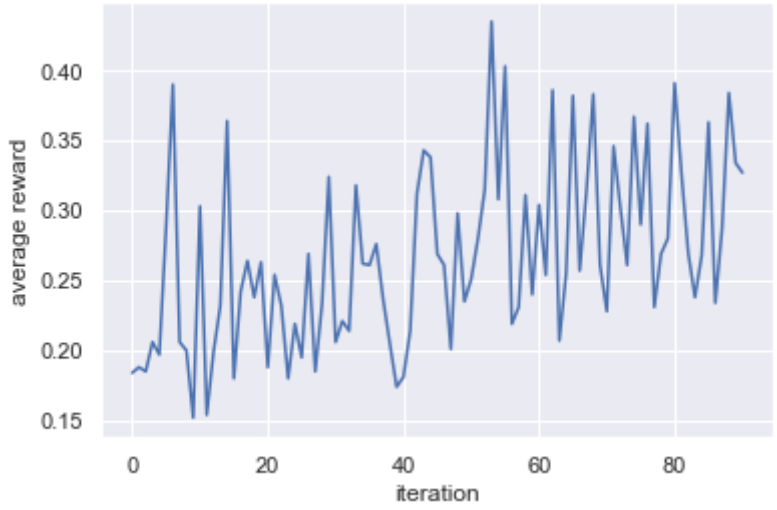


Average length:

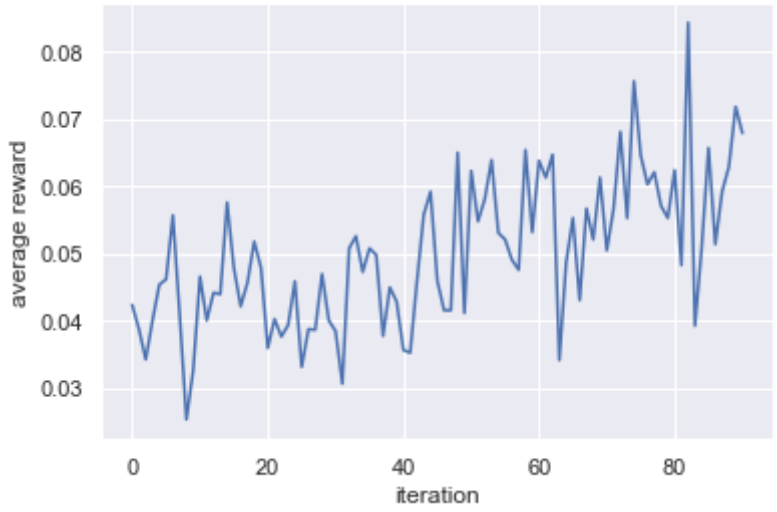


- Setting 2 ($r_t = s * \gamma^{T-t}$)

Cumulative reward:



Final reward:



Update (5-5)

Next step:

1. Use the setting without intermediate reward.
2. Change the reward of "no modification" action (-0.5, 0, 0.5?)
3. Use the setting with intermediate reward first and then

Discussion:

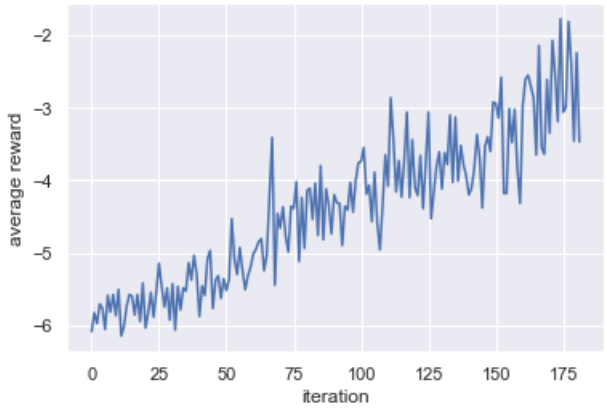
1. Given any allele, we want to generate a database of peptides that can trigger immune response.
2. Given any allele, we want to generate a database of peptides that can target the virus genome.

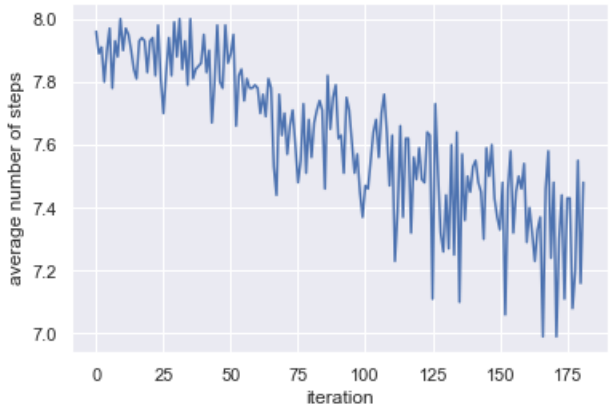
Title:

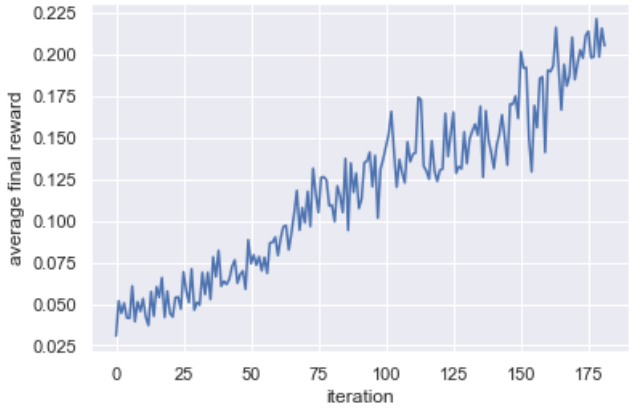
peptide mutation policy with combinatorial constraint (i.e., mutate the peptide sequences with the constraint of amino acid difference).

- Setting 1 (+1, -1, -0.9)

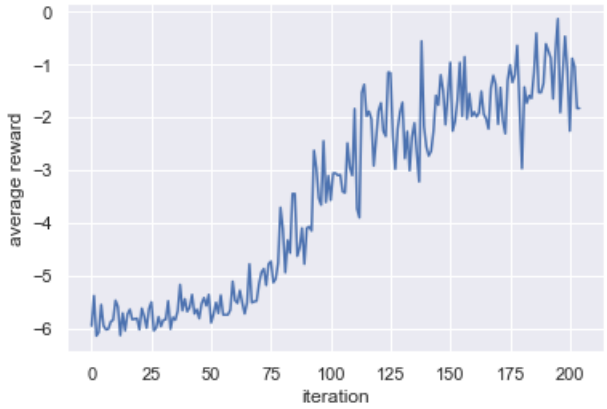
- Without warm start (dictionary)

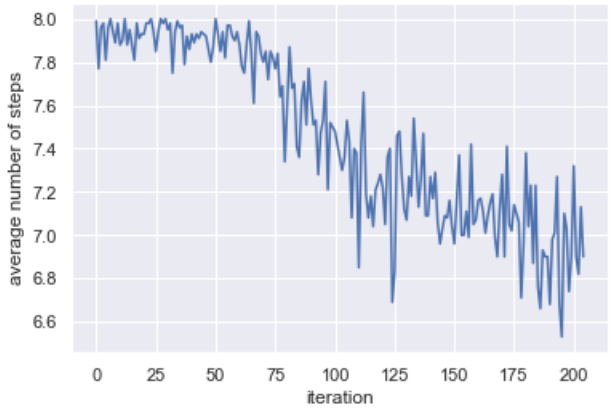


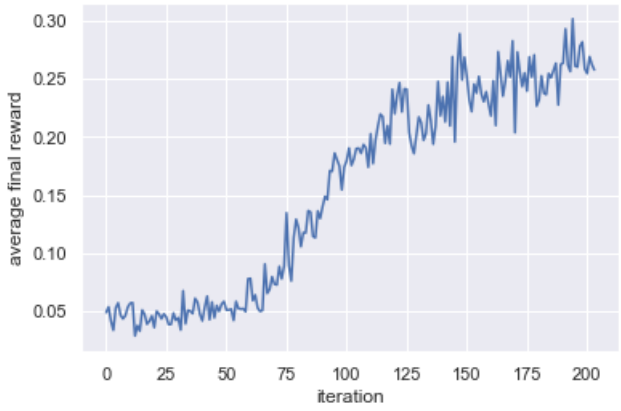




- With warm start





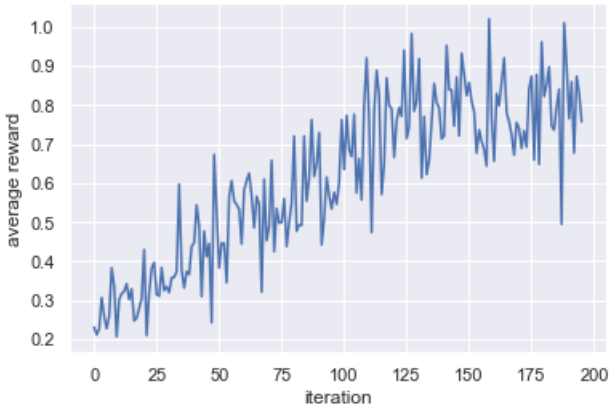


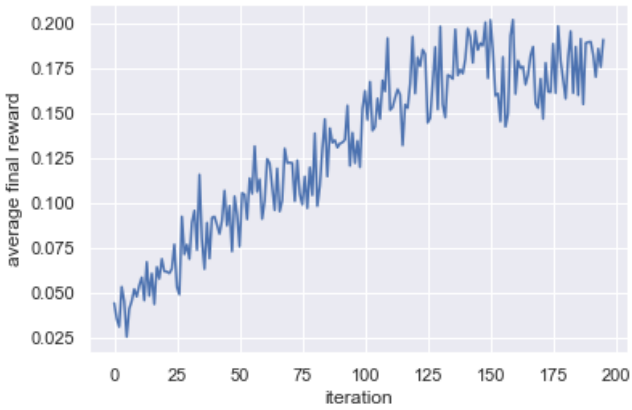
Example:

> terminal: False; action: 0,1,19; old_peptide: VTYLALLAAF; new_peptides: VYYLALLAAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: -1.0000; process_score: 0.0331; present_score: 0.0047; affinity: 0.0688;
>
> terminal: False; action: 0,0,19; old_peptide: VYYLALLAAF; new_peptides: YYYLALLAAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: -1.0000; process_score: 0.0447; present_score: 0.0048; affinity: 0.0656;
>
> terminal: False; action: 0,4,15; old_peptide: YYYLALLAAF; new_peptides: YYYLPLLAAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: -1.0000; process_score: 0.1511; present_score: 0.0074; affinity: 0.0712;
>
> terminal: False; action: 0,3,15; old_peptide: YYYLPLLAAF; new_peptides: YYYPPLLAAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: 1.0000; process_score: 0.7602; present_score: 0.1436; affinity: 0.1628;
>
> terminal: False; action: 0,5,15; old_peptide: YYYPPLLAAF; new_peptides: YYYPPPLAAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: 1.0000; process_score: 0.9098; present_score: 0.3412; affinity: 0.2197;

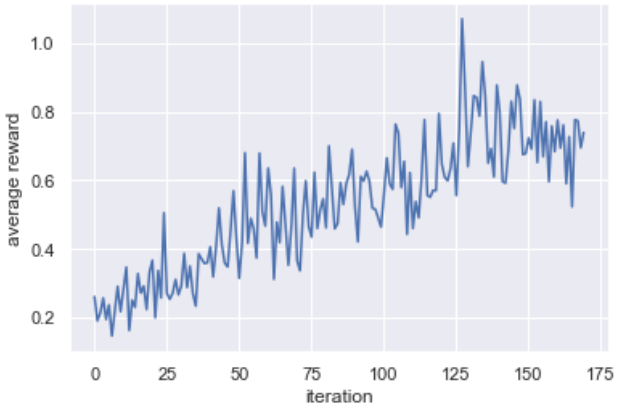
> terminal: False; action: 0,6,15; old_peptide: YYYPPPLAAF; new_peptides: YYYPPPPAAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: 1.0000; process_score: 0.8753; present_score: 0.5267; affinity: 0.3041;
>
> terminal: False; action: 0,7,15; old_peptide: YYYPPPPAAF; new_peptides: YYYPPPPPAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: -1.0000; process_score: 0.8238; present_score: 0.4826; affinity: 0.3046;
>
> terminal: True; action: 0,2,15; old_peptide: YYYPPPPPAF; new_peptides: YYPPPPPPAF; allele: YFAMYEESAAHTDVDTLYIIYRDYTWAARAYTWY; rewards: 3.5450; process_score: 0.4992; present_score: 0.3545; affinity: 0.3635;

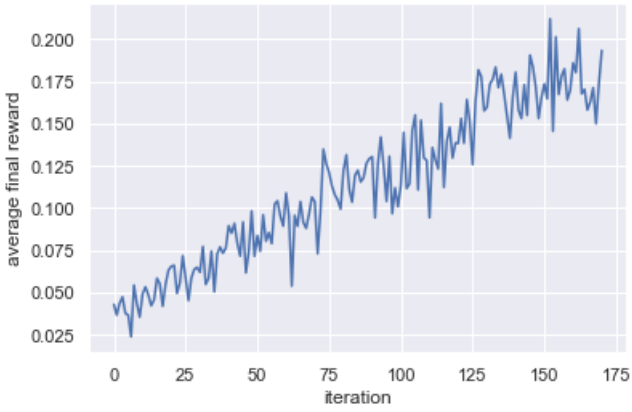
- Setting 2 ($r_t = s * \gamma^{T-t}$)
  - Without warm start (dictionary)





  - With warm start





terminal: False; action: 0,8,20; old_peptide: GRMEKKTWK; new_peptides: GRMEKKTWV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.0058; process_score: 0.2096; present_score: 0.0121; affinity: 0.0990;

terminal: False; action: 0,7,20; old_peptide: GRMEKKTWV; new_peptides: GRMEKKTVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.0191; process_score: 0.3005; present_score: 0.0359; affinity: 0.1745;

terminal: False; action: 0,6,9; old_peptide: GRMEKKTVV; new_peptides: GRMEKKHVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.0131; process_score: 0.1694; present_score: 0.0222; affinity: 0.1715;

terminal: False; action: 0,1,20; old_peptide: GRMEKKHVV; new_peptides: GVMEKKHVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.0352; process_score: 0.2614; present_score: 0.0536; affinity: 0.2274;

terminal: False; action: 0,4,9; old_peptide: GVMEKKHVV; new_peptides: GVMEHKHVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.0510; process_score: 0.2005; present_score: 0.0699; affinity: 0.2748;

terminal: False; action: 0,4,20; old_peptide: GVMEHKHVV; new_peptides: GVMEVKHVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.1047; process_score: 0.4881; present_score: 0.1292; affinity: 0.2428;

terminal: False; action: 0,5,20; old_peptide: GVMEVKHVV; new_peptides: GVMEVVHVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.5094; process_score: 0.9276; present_score: 0.5660; affinity: 0.3016;

terminal: True; action: 0,0,9; old_peptide: GVMEVVHVV; new_peptides: HVMEVVHVV; allele: YYTKYREISTNTYESNLYWRYNLYTWAELAYLWY; rewards: 0.6848; process_score: 0.9049; present_score: 0.6848; affinity: 0.3578;

Update (5-12)

Suggestions:

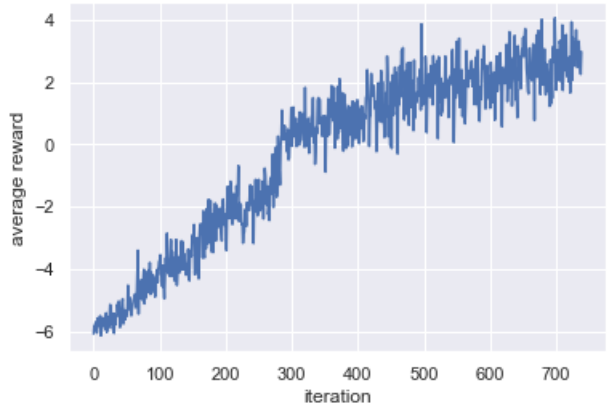
1. Use a larger episode length
2. Set the threshold as 0.75

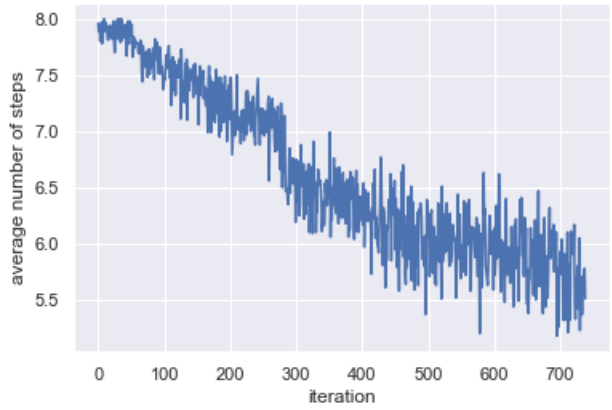
1. Result Analysis with longer running time

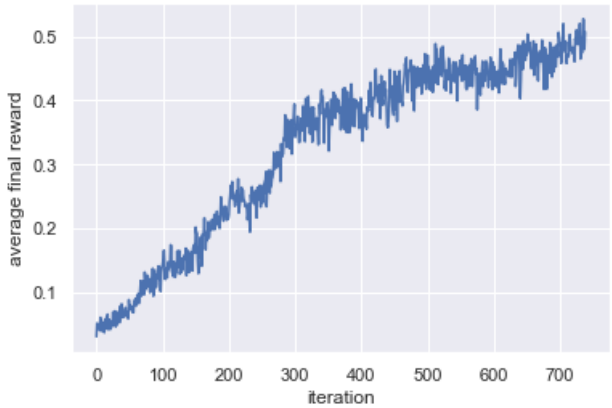
- Setting 1 (+1, -1, -0.9) without warm start (i.e., prior distribution dictionary)

  Among all the samples (i.e., the initial state of each episode/trajectory), 80% of them are sampled from the IEDB dataset; 20% from randomly generated peptides.

  It takes 60 hours to get the results below.

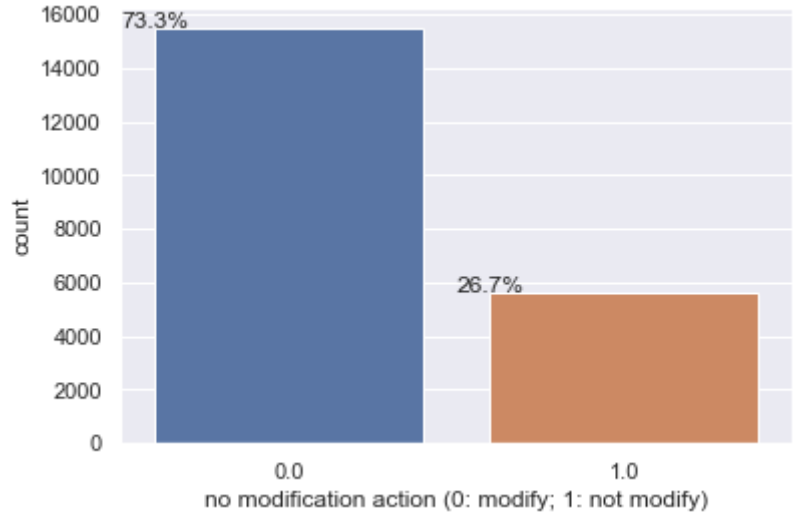


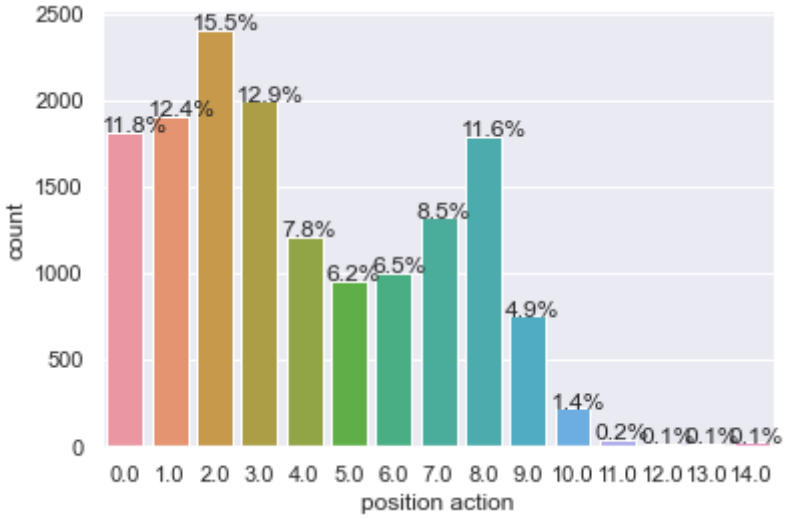


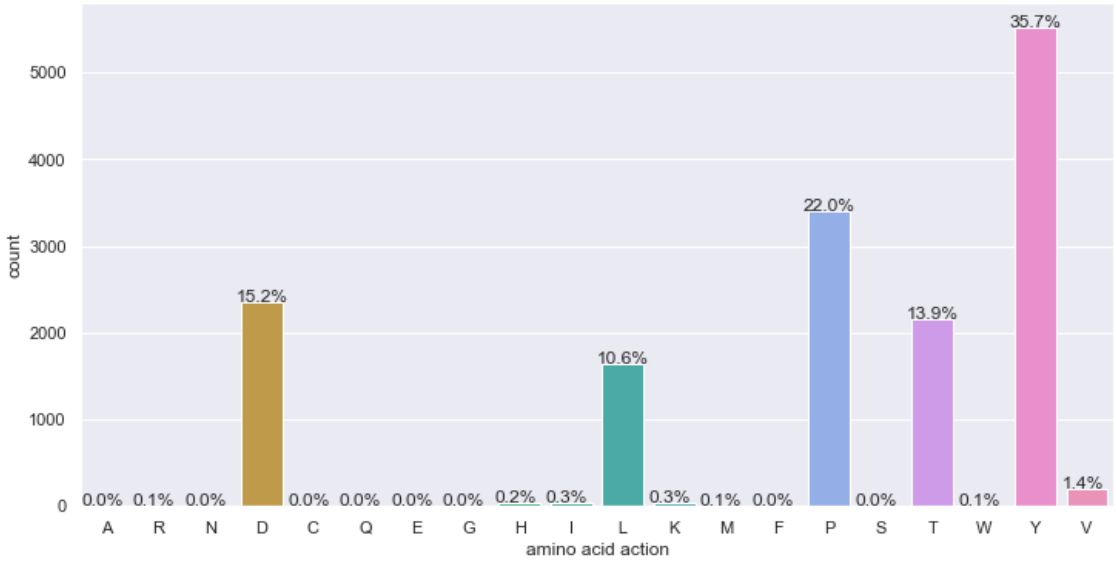


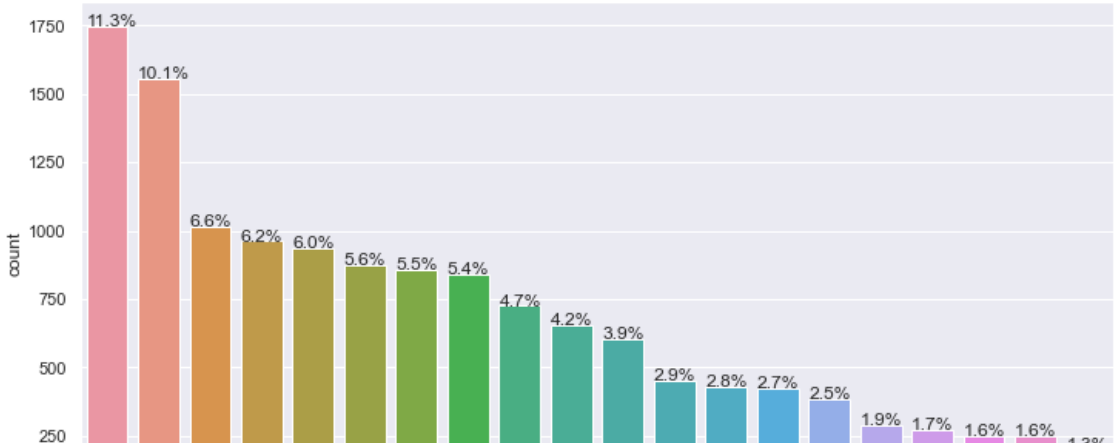
Each iteration has 2,112 steps. In each step, a specific given state will be updated by the policy network. Within 739 iterations, we have 238,343 samples in total. In order to analyze the trained policy network, we select the corresponding optimized peptides (i.e., the peptides in the terminal state) of the last 3,750 (about 5%) samples in the last 10 iterations for further analysis.

- Analysis 1: action analysis

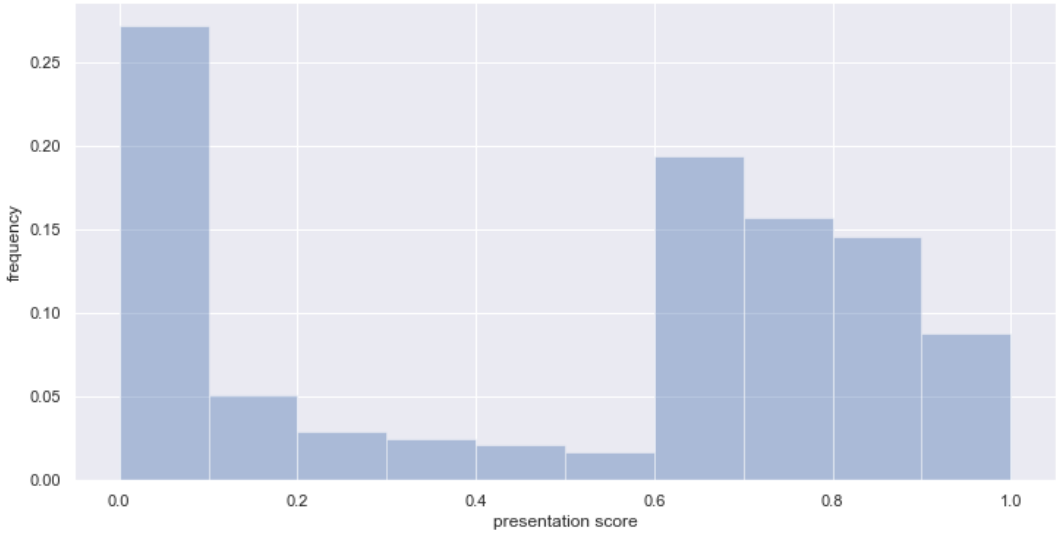

16000
14000
12000
10000
8000
6000
4000
2000
0
count
73.3%
26.7%
0.0
1.0
no modification action (0: modify; 1: not modify)


2500
2000
1500
1000
500
0
count
11.8%
12.4%
15.5%
12.9%
7.8%
6.2%
6.5%
8.5%
11.6%
4.9%
1.4%
0.2%
0.1%
0.1%
0.1%
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0
position action

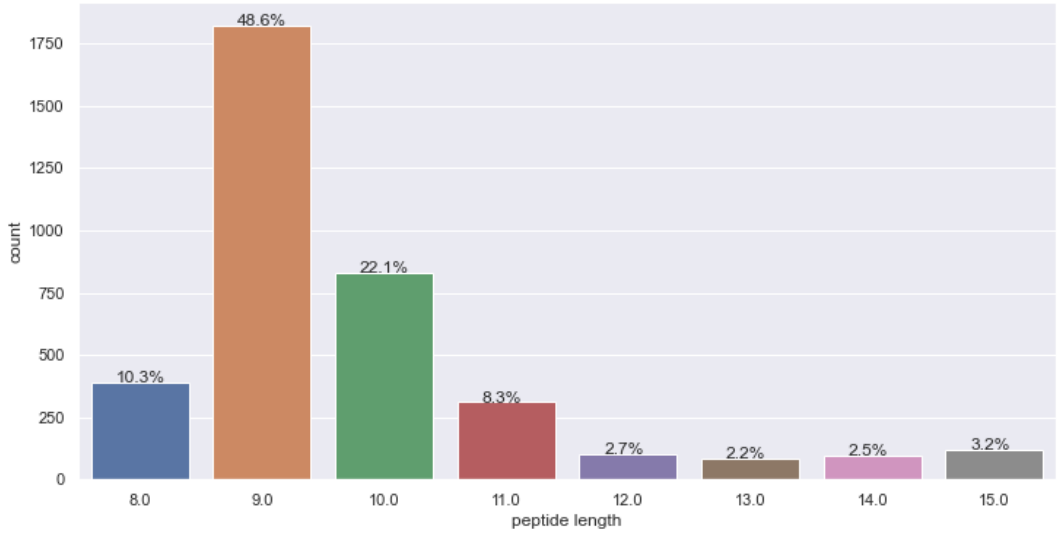

5000
4000
3000
2000
1000
0
count
0.0% 0.1% 0.0% 15.2% 0.0% 0.0% 0.0% 0.0% 0.2% 0.3% 10.6% 0.3% 0.1% 0.0% 22.0% 0.0% 13.9% 0.1% 35.7% 1.4%
A R N D C Q E G H I L K M F P S T W Y V
amino acid action

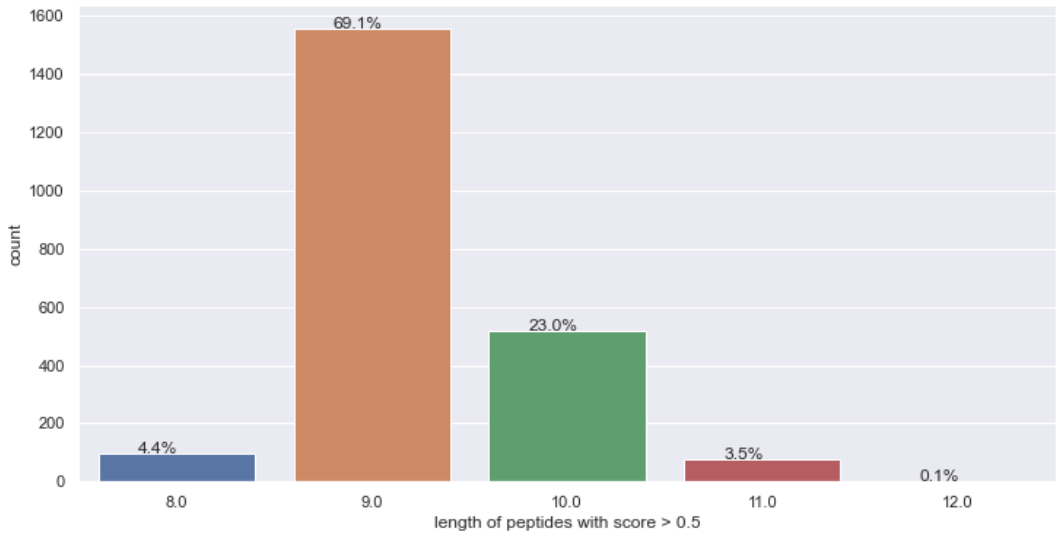

1750
1500
1250
1000
750
500
250
count
11.3%
10.1%
6.6%
6.2%
6.0%
5.6%
5.5%
5.4%
4.7%
4.2%
3.9%
2.9%
2.8%
2.7%
2.5%
1.9%
1.7%
1.6%
1.6%

- Analysis 2: Reward distribution (60.16% optimized peptides have the presentation scores > 0.5)



- Analysis 3: peptide length distribution





Examples:

> 1. Episodes with many "no modification" actions

terminal: False; action: 0,7,11; old_peptide: WGCCPEVS; new_peptides: WGCCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -1.0000; process_score: 0.0054; present_score: 0.0059; affinity: 0.0984;
terminal: False; action: 1,0,1; old_peptide: WGCCPEVL; new_peptides: WGCCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -0.9000; process_score: 0.0054; present_score: 0.0059; affinity: 0.0984;
terminal: False; action: 1,0,1; old_peptide: WGCCPEVL; new_peptides: WGCCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -0.9000; process_score: 0.0054; present_score: 0.0059; affinity: 0.0984;
terminal: False; action: 1,0,1; old_peptide: WGCCPEVL; new_peptides: WGCCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -0.9000; process_score: 0.0054; present_score: 0.0059; affinity: 0.0984;
terminal: False; action: 1,0,1; old_peptide: WGCCPEVL; new_peptides: WGCCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -0.9000; process_score: 0.0054; present_score: 0.0059; affinity: 0.0984;
terminal: False; action: 0,2,15; old_peptide: WGCCPEVL; new_peptides: WGPCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -1.0000; process_score: 0.0208; present_score: 0.0148; affinity: 0.1822;
terminal: False; action: 1,0,1; old_peptide: WGPCPEVL; new_peptides: WGPCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: -0.9000; process_score: 0.0208; present_score: 0.0148; affinity: 0.1822;
terminal: True; action: 1,0,1; old_peptide: WGPCPEVL; new_peptides: WGPCPEVL; allele: YYAGYREKYRQTDVSNLYIRYDYYTWAEWAYLWY; rewards: 0.1481; process_score: 0.0208; present_score: 0.0148; affinity: 0.1822;

---

terminal: False; action: 1,0,1; old_peptide: PVINVQDL; new_peptides: PVINVQDL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -0.9000; process_score: 0.0246; present_score: 0.0039; affinity: 0.0524;
terminal: False; action: 1,0,1; old_peptide: PVINVQDL; new_peptides: PVINVQDL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -0.9000; process_score: 0.0246; present_score: 0.0039; affinity: 0.0524;
terminal: False; action: 0,6,17; old_peptide: PVINVQDL; new_peptides: PVINVQTL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -1.0000; process_score: 0.1038; present_score: 0.0055; affinity: 0.0589;
terminal: False; action: 1,0,1; old_peptide: PVINVQTL; new_peptides: PVINVQTL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -0.9000; process_score: 0.1038; present_score: 0.0055; affinity: 0.0589;
terminal: False; action: 0,1,15; old_peptide: PVINVQTL; new_peptides: PPINVQTL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -1.0000; process_score: 0.0141; present_score: 0.0038; affinity: 0.0546;
terminal: False; action: 0,0,19; old_peptide: PPINVQTL; new_peptides: YPINVQTL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -1.0000; process_score: 0.4524; present_score: 0.0213; affinity: 0.0720;
terminal: False; action: 0,3,4; old_peptide: YPINVQTL; new_peptides: YPIDVQTL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: -1.0000; process_score: 0.3719; present_score: 0.0181; affinity: 0.0836;
terminal: True; action: 0,5,17; old_peptide: YPIDVQTL; new_peptides: YPIDVTTL; allele: YFAMYGEKVAHTHVDTLYVRYQYYTWAVLAYTWY; rewards: 0.1499; process_score: 0.3121; present_score: 0.0150; affinity: 0.0853;

2. Examples with large final rewards

terminal: False; action: 0,0,19; old_peptide: KEEESEGAIW; new_peptides: YEEESEGAIW; allele: YYAEYRNIYAQTDESNLYIRYDSYTWAVLAYLWY; rewards: -1.0000; process_score: 0.3170; present_score: 0.0973; affinity: 0.2699;
terminal: False; action: 0,2,19; old_peptide: YEEESEGAIW; new_peptides: YEYESEGAIW; allele: YYAEYRNIYAQTDESNLYIRYDSYTWAVLAYLWY; rewards: 1.0000; process_score: 0.7595; present_score: 0.5092; affinity: 0.3364;
terminal: False; action: 0,9,11; old_peptide: YEYESEGAIW; new_peptides: YEYESEGAIL; allele: YYAEYRNIYAQTDESNLYIRYDSYTWAVLAYLWY; rewards: -1.0000; process_score: 0.6002; present_score: 0.4058; affinity: 0.3502;
terminal: False; action: 0,1,19; old_peptide: YEYESEGAIL; new_peptides: YYYESEGAIL; allele: YYAEYRNIYAQTDESNLYIRYDSYTWAVLAYLWY; rewards: 1.0000; process_score: 0.6715; present_score: 0.4866; affinity: 0.3574;
terminal: True; action: 0,3,4; old_peptide: YYYESEGAIL; new_peptides: YYYDSEGAIL; allele: YYAEYRNIYAQTDESNLYIRYDSYTWAVLAYLWY; rewards: 7.7289; process_score: 0.7591; present_score: 0.7729; affinity: 0.4496;

---

terminal: False; action: 0,8,11; old_peptide: STLAGVNVG; new_peptides: STLAGVNVL; allele: YYSKYRNIYAQTDESNLYLSYDYYTWAERAYEWY; rewards: -1.0000; process_score: 0.1842; present_score: 0.0533; affinity: 0.2529;
terminal: False; action: 0,3,15; old_peptide: STLAGVNVL; new_peptides: STLPGVNVL; allele: YYSKYRNIYAQTDESNLYLSYDYYTWAERAYEWY; rewards: 1.0000; process_score: 0.3234; present_score: 0.2261; affinity: 0.3626;
terminal: True; action: 0,2,4; old_peptide: STLPGVNVL; new_peptides: STDPGVNVL; allele: YYSKYRNIYAQTDESNLYLSYDYYTWAERAYEWY; rewards: 7.3298; process_score: 0.7001; present_score: 0.7330; affinity: 0.4490;

3. Examples with final presentation score < 0.5

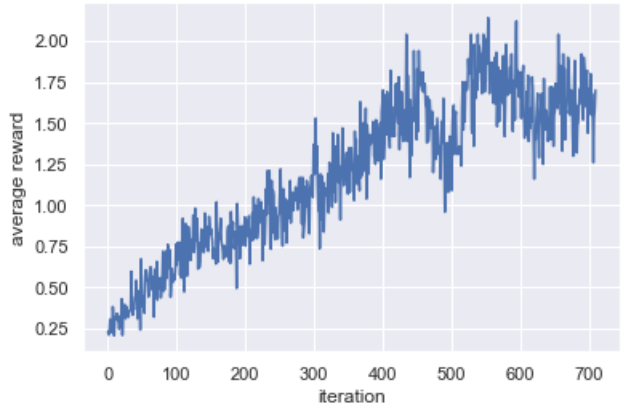
Problem: how to encourage examples like this to select as many "no modification" actions as possible?

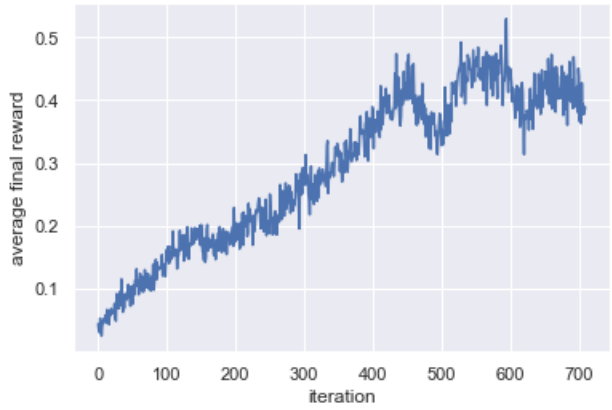
terminal: False; action: 0,4,15; old_peptide: TLVQYMDDI; new_peptides: TLVQPMDDI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: -1.0000; process_score: 0.0111; present_score: 0.0044; affinity: 0.0687;
terminal: False; action: 0,6,17; old_peptide: TLVQPMDDI; new_peptides: TLVQPMTDI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: -1.0000; process_score: 0.0426; present_score: 0.0052; affinity: 0.0740;
terminal: False; action: 0,7,17; old_peptide: TLVQPMTDI; new_peptides: TLVQPMTTI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: -1.0000; process_score: 0.2484; present_score: 0.0133; affinity: 0.0952;
terminal: False; action: 0,3,15; old_peptide: TLVQPMTTI; new_peptides: TLVPPMTTI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: -1.0000; process_score: 0.1736; present_score: 0.0123; affinity: 0.1131;
terminal: False; action: 0,2,4; old_peptide: TLVPPMTTI; new_peptides: TLDPPMTTI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: -1.0000; process_score: 0.5161; present_score: 0.0623; affinity: 0.1569;
terminal: False; action: 0,5,15; old_peptide: TLDPPMTTI; new_peptides: TLDPPPTTI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: 1.0000; process_score: 0.6149; present_score: 0.1562; affinity: 0.2211;
terminal: False; action: **0,1,19**; old_peptide: TLDPPPTTI; new_peptides: TYDPPPTTI; allele:

> YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: -1.0000; process_score: 0.5693; present_score: 0.1088; affinity: 0.1969;
> terminal: True; action: 0,1,4; old_peptide: TYDPPPTTI; new_peptides: TDDPPPTTI; allele: YHTKYREISTNTYESNLYWRYNLHTWAELAYLWY; rewards: 3.1405; process_score: 0.1116; present_score: 0.3140; affinity: 0.4767;

- Setting 2 ($r_t = s * \gamma^{T-t}$) without warm start (dictionary)

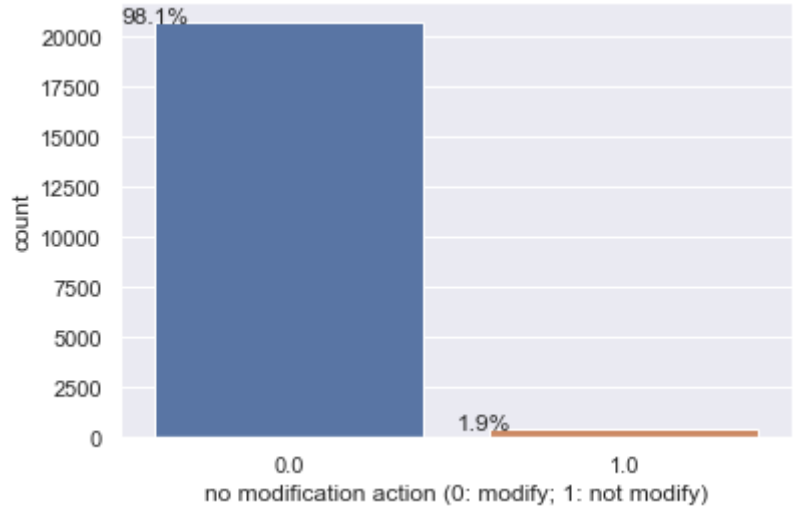
  Among all the samples (i.e., the initial state of each episode/trajectory), 80% of them are sampled from the IEDB dataset; 20% from randomly generated peptides.

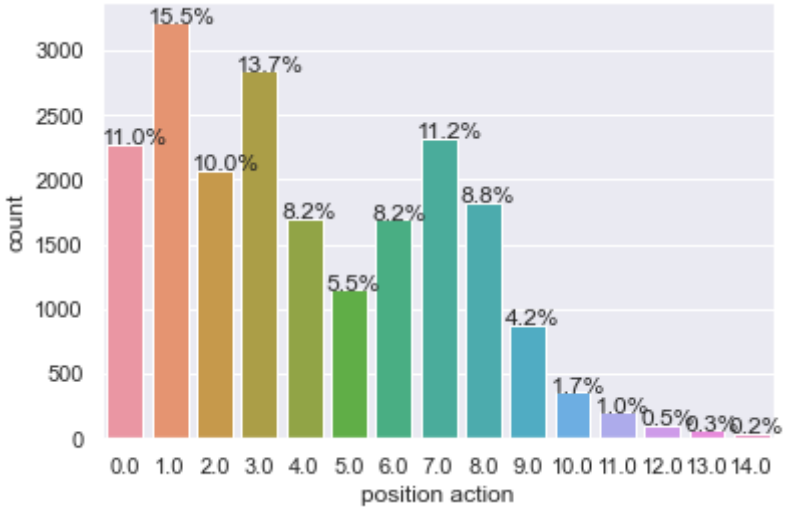


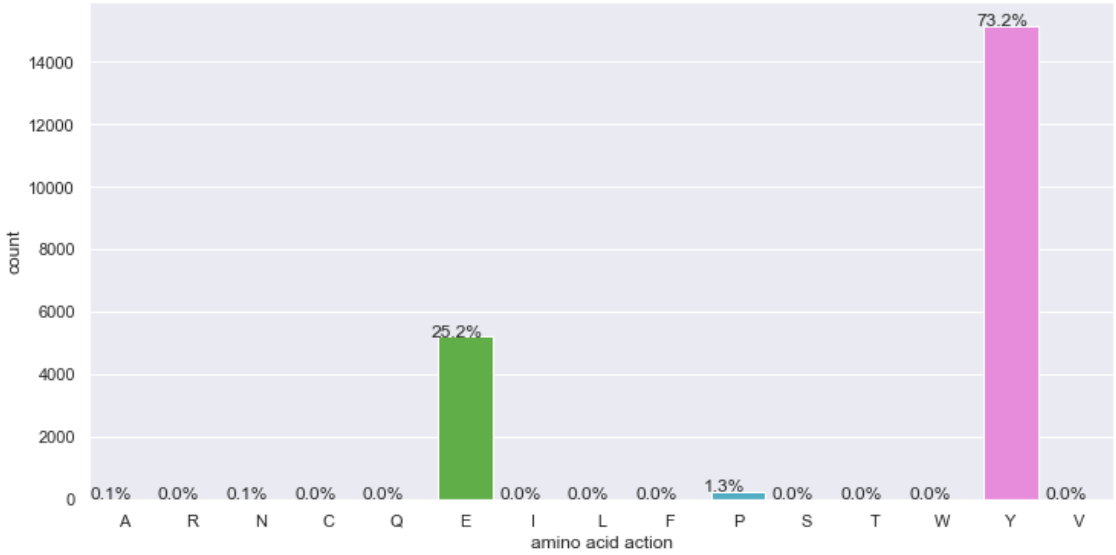


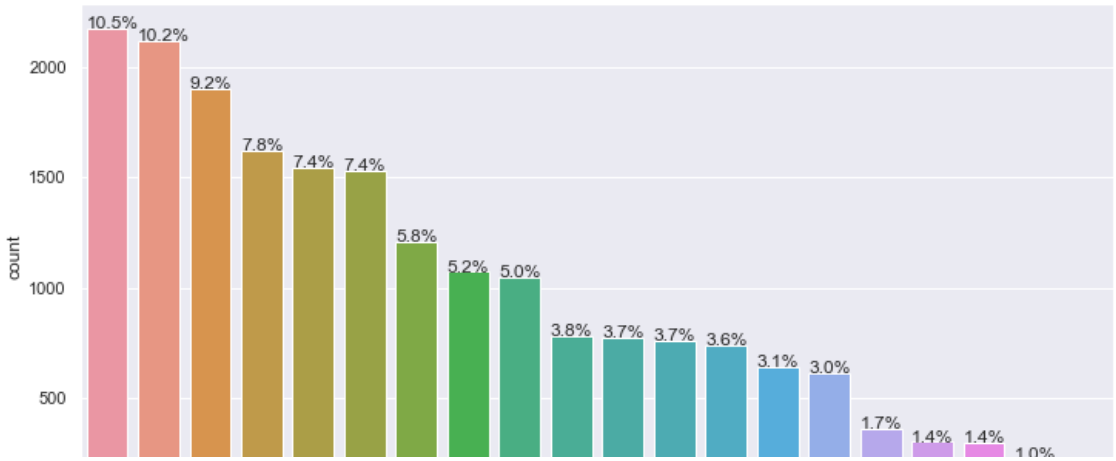
  Each iteration has 2,112 steps. In each step, a specific given state will be updated by the policy network. Within 710 iterations, we have 187,440 samples in total. In order to analyze the trained policy network, we select the corresponding optimized peptides (i.e., the peptides in the terminal state) of the last 2,640 (about 1.4%) samples in the last 10 iterations for further analysis.

  - Analysis 1: action analysis

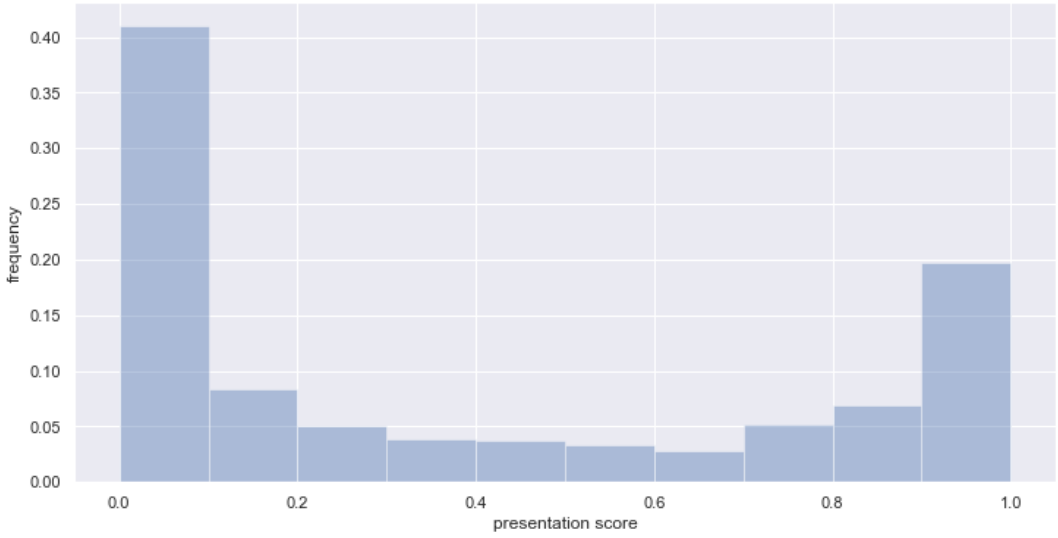

98.1%
1.9%
count
0.0
1.0
no modification action (0: modify; 1: not modify)


11.0%
15.5%
10.0%
13.7%
8.2%
5.5%
8.2%
11.2%
8.8%
4.2%
1.7%
1.0%
0.5%
0.3%
0.2%
count
position action

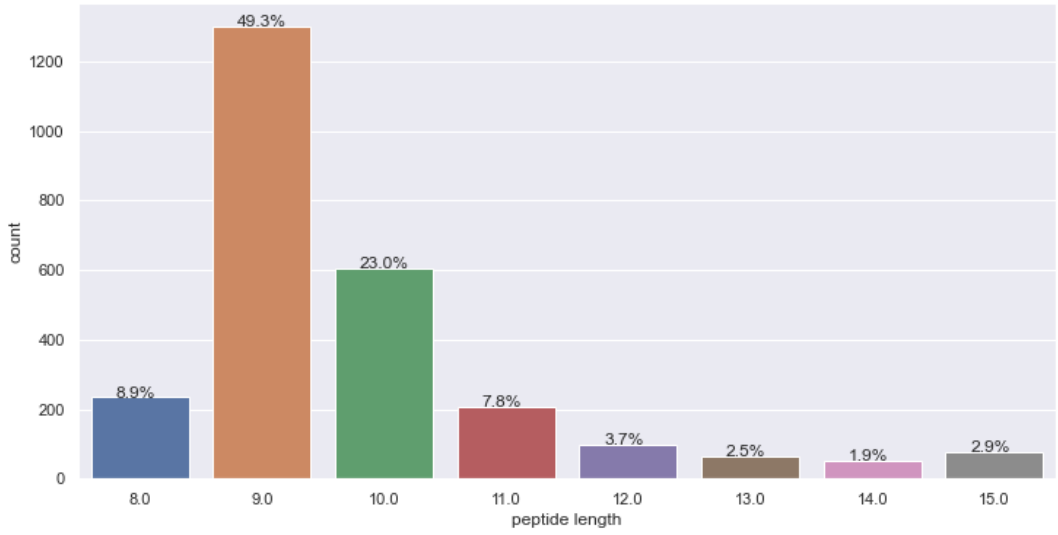

0.1%
0.0%
0.1%
0.0%
0.0%
25.2%
0.0%
0.0%
0.0%
1.3%
0.0%
0.0%
0.0%
73.2%
0.0%
A
R
N
C
Q
E
I
L
F
P
S
T
W
Y
V
count
amino acid action

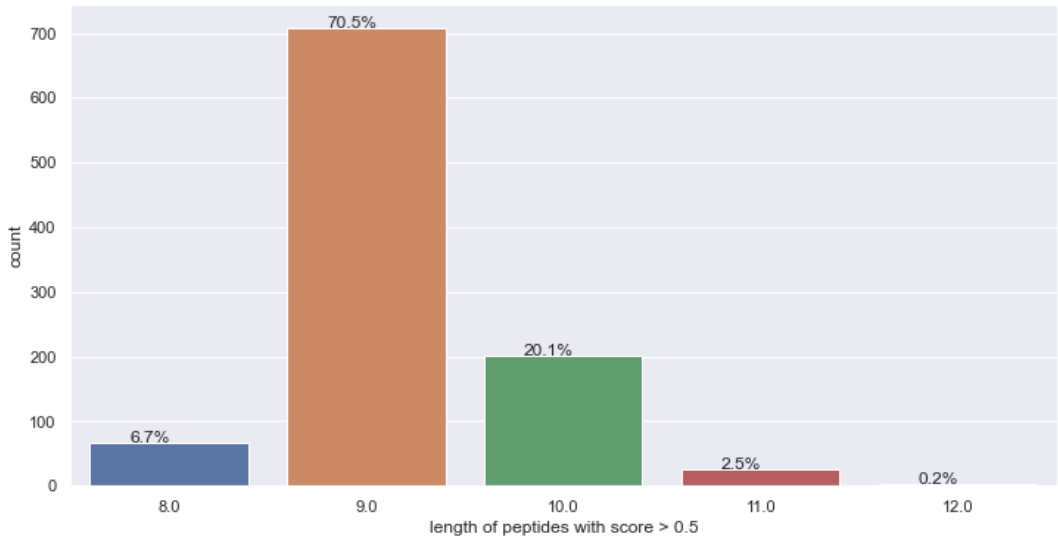

10.5%
10.2%
9.2%
7.8%
7.4%
7.4%
5.8%
5.2%
5.0%
3.8%
3.7%
3.7%
3.6%
3.1%
3.0%
1.7%
1.4%
1.4%
count

- Analysis 2: Reward distribution (38.07% optimized peptides have the presentation scores > 0.5)



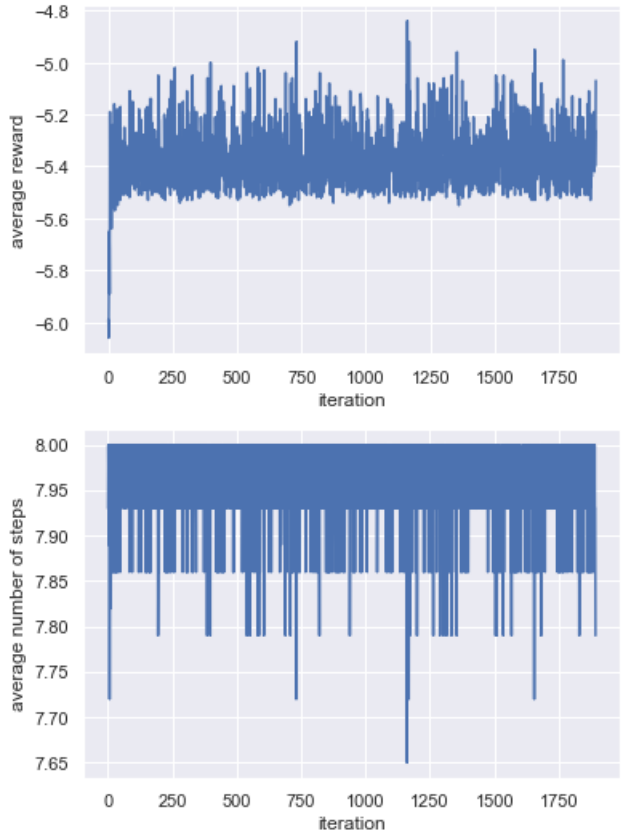
- Analysis 3: peptide length distribution



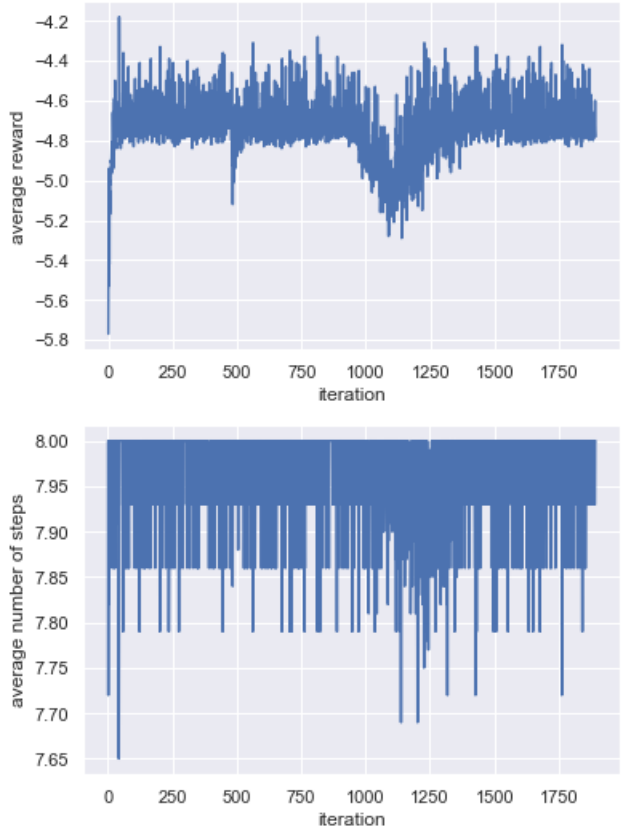


2. Result Analysis with smaller penalty (i.e., greater reward) for "no modification" action.

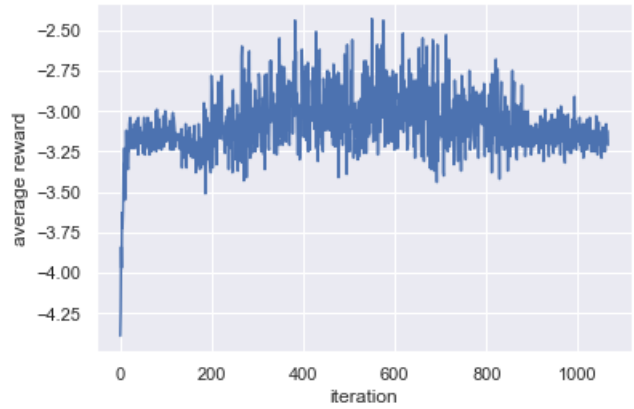
- Setting 1 (+1, -1, -0.8) without warm start

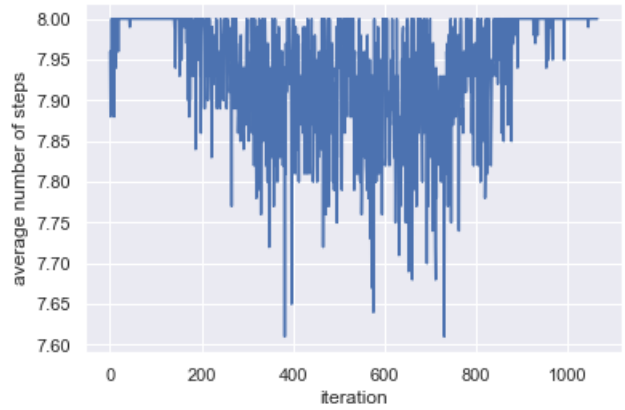


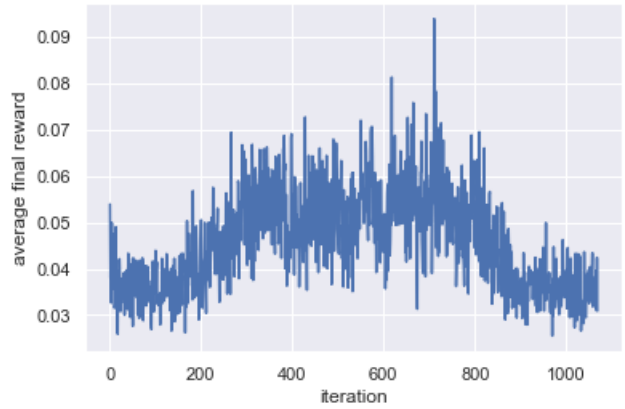
- Setting 1 (+1, -1, -0.7) without warm start



- Setting 1 (+1, -1, -0.5) without warm start







3. Result Analysis with new action spaces

Question:

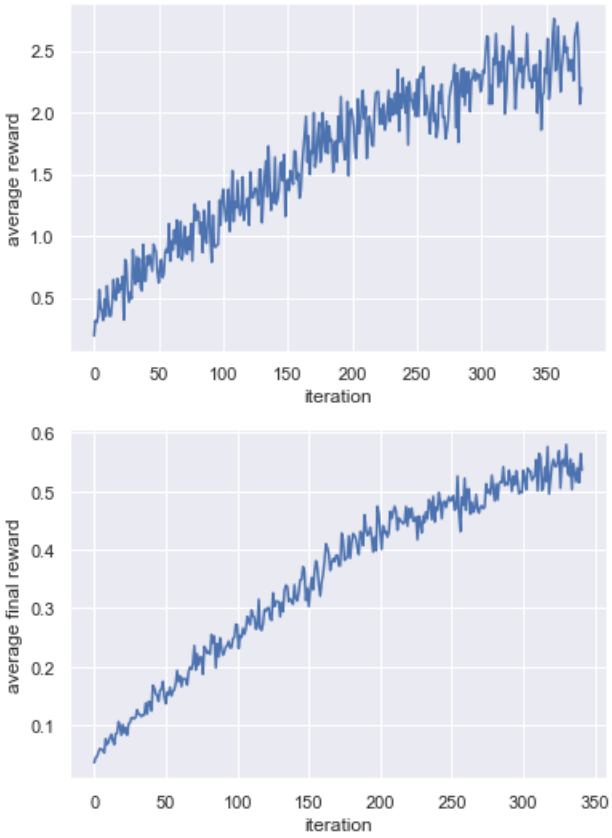
The reason why the training of model fails with penalty greater than -0.9 could be that, with current action space, if we select "no modification" action, our model cannot update the network for position prediction and amino acid prediction. With small penalty for "no modification" action, the training of our model can get stuck at the "no modification" action and don't update the other two networks.

Therefore, based on this assumption, I wonder whether combining the "no modification" action with the "position" action can help alleviate this issue. With this new action space,

$R = \sum \gamma^t r_t$

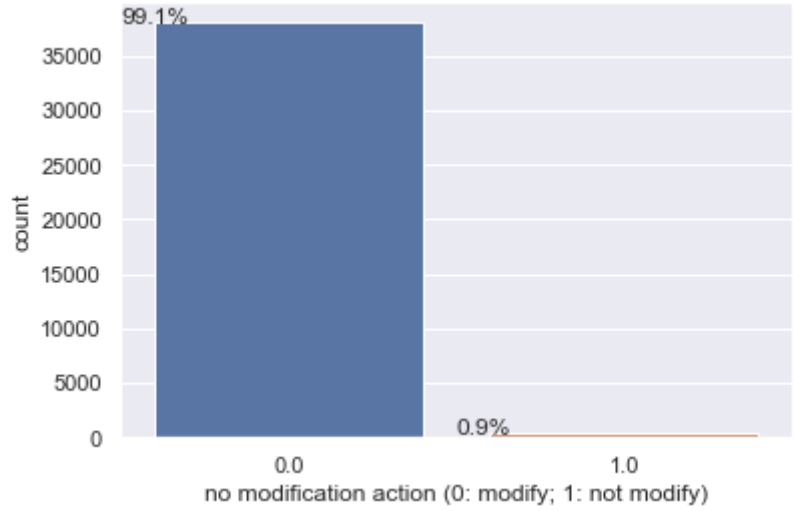
- Setting 2 ($r_t = s_t * \alpha^{T-t} + \beta$) with two actions (position & no modification, amino acid).

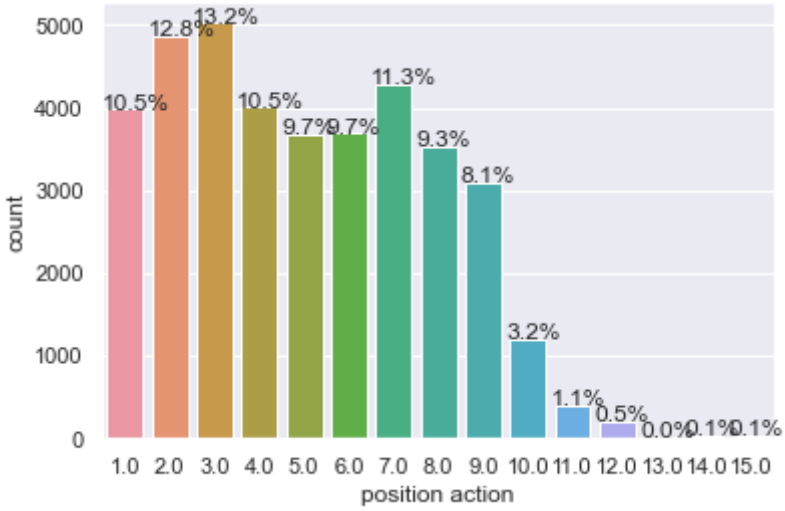
  Among all the samples (i.e., the initial state of each episode/trajectory), 80% of them are sampled from the IEDB dataset; 20% from randomly generated peptides.

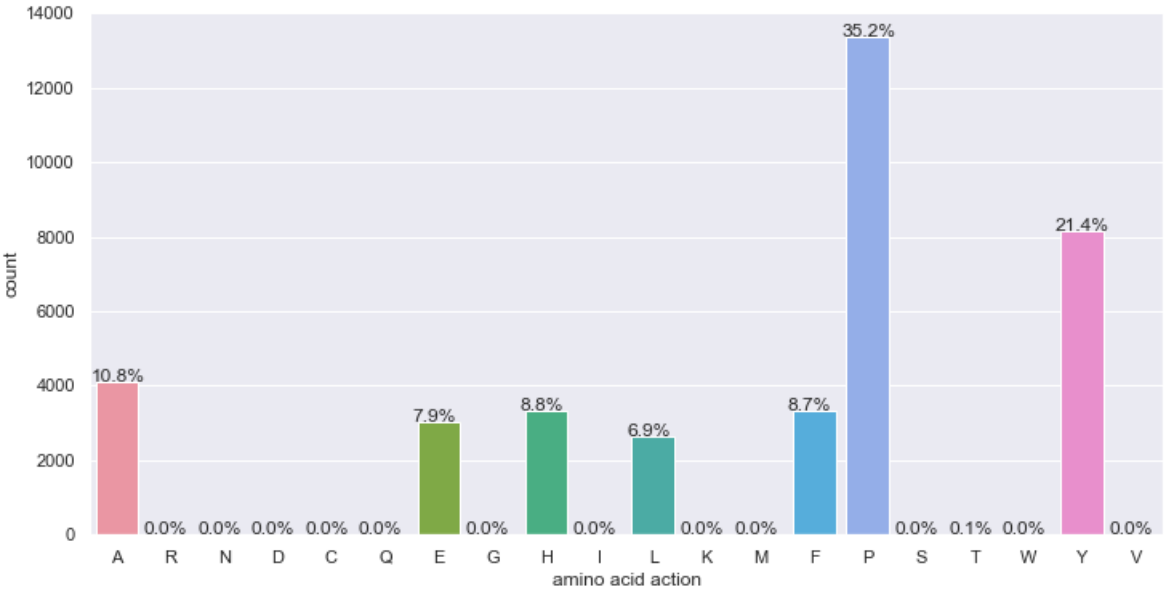


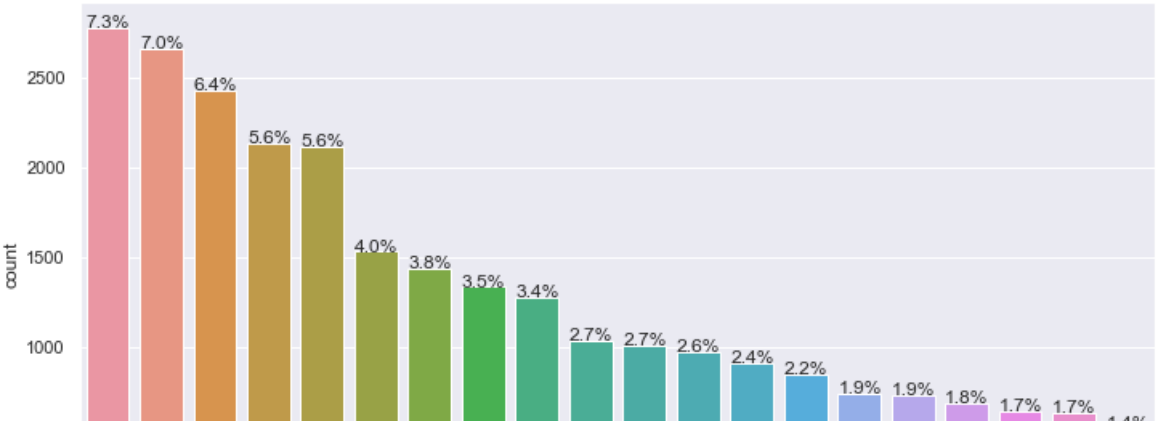
Each iteration has 3,840 steps. In each step, a specific given state will be updated by the policy network. Within 378 iterations, we have 181,440 samples in total. In order to analyze the trained policy network, we select the corresponding optimized peptides (i.e., the peptides in the terminal state) of the last 4,800 (about 2.6%) samples in the last 10 iterations for further analysis.

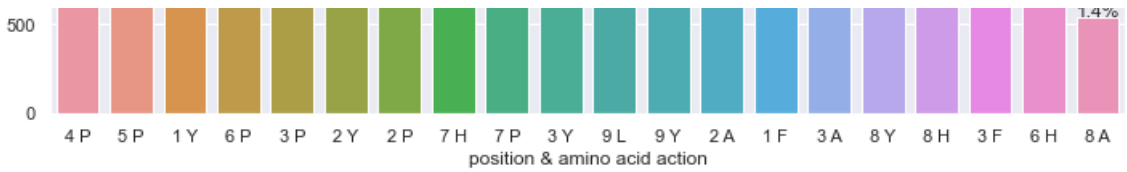
- Analysis 1: action analysis


99.1%
0.9%
count
0.0
1.0
no modification action (0: modify; 1: not modify)

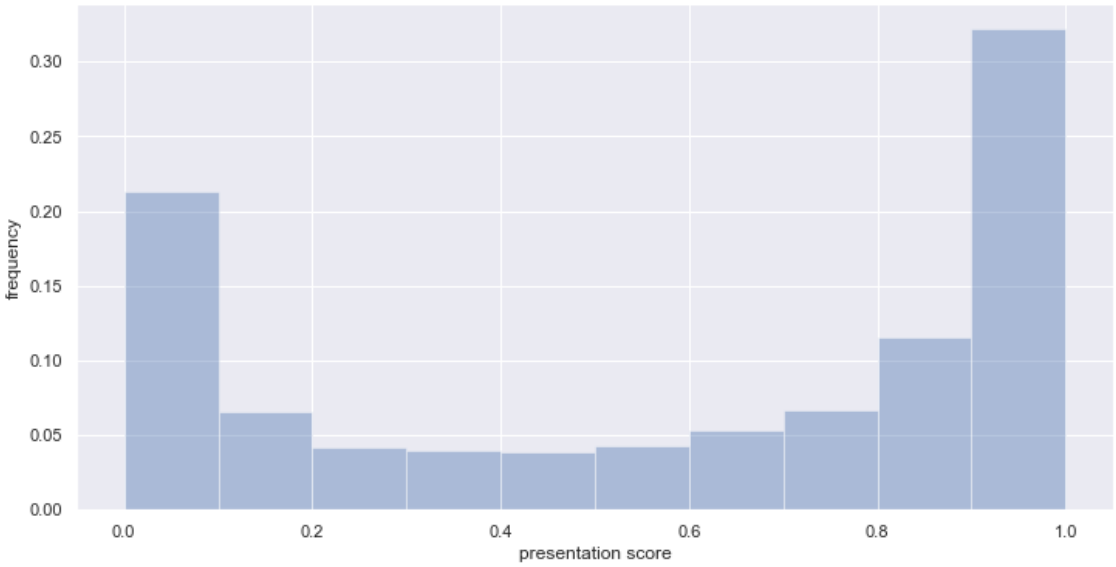

10.5%
12.8%
13.2%
10.5%
9.7%
9.7%
11.3%
9.3%
8.1%
3.2%
1.1%
0.5%
0.0%
0.1%
0.1%
count
position action


10.8%
0.0%
0.0%
0.0%
0.0%
0.0%
7.9%
0.0%
8.8%
0.0%
6.9%
0.0%
0.0%
8.7%
35.2%
0.0%
0.1%
0.0%
21.4%
0.0%
A R N D C Q E G H I L K M F P S T W Y V
count
amino acid action

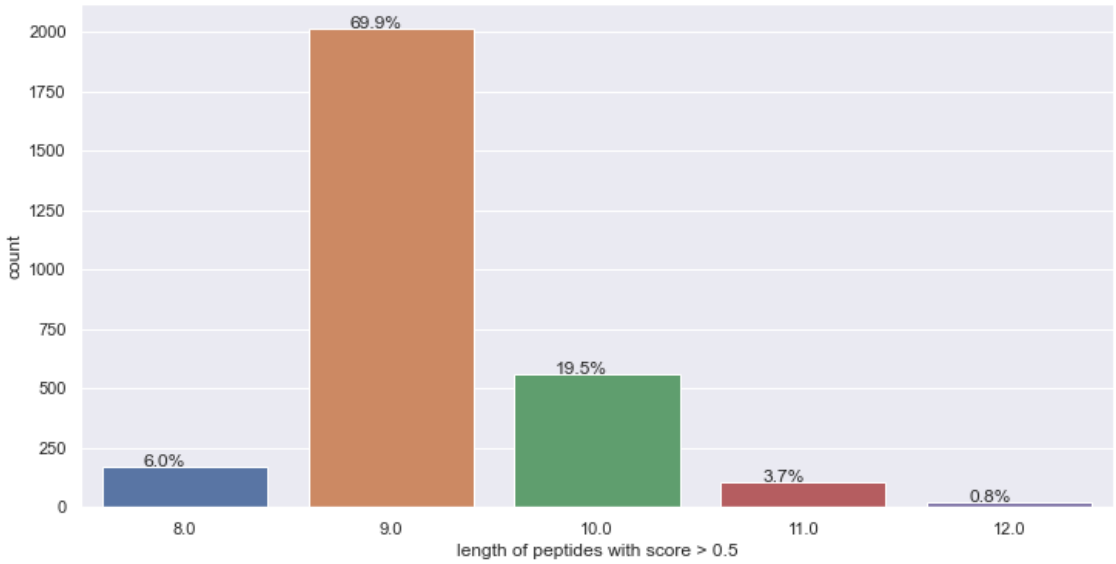

7.3%
7.0%
6.4%
5.6%
5.6%
4.0%
3.8%
3.5%
3.4%
2.7%
2.7%
2.6%
2.4%
2.2%
1.9%
1.9%
1.8%
1.7%
1.7%
count

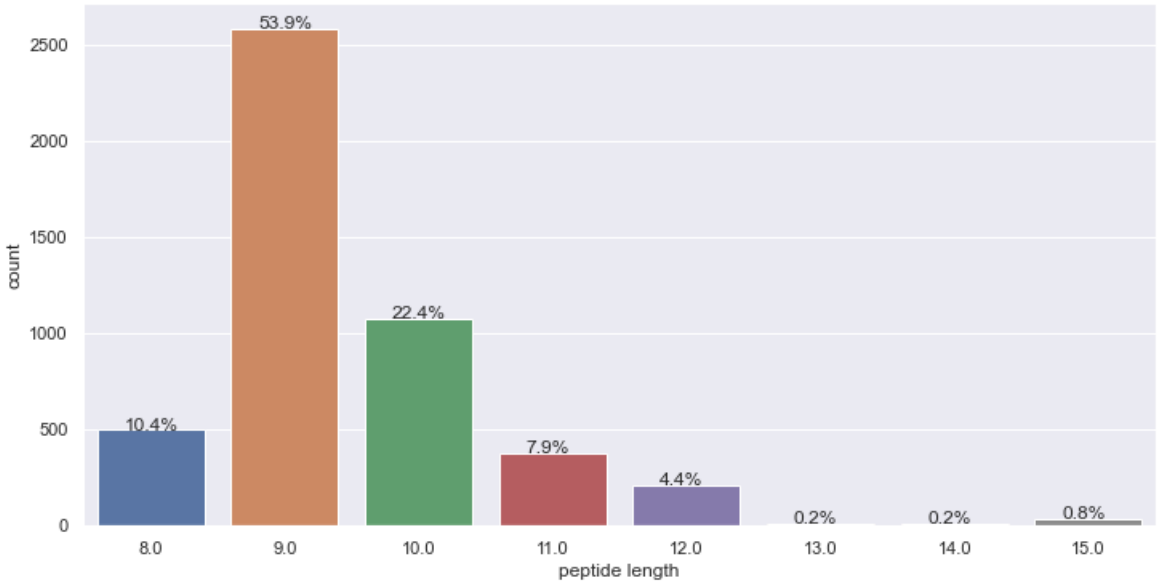


- Analysis 2: Reward distribution (60.08% optimized peptides have the presentation scores > 0.5)



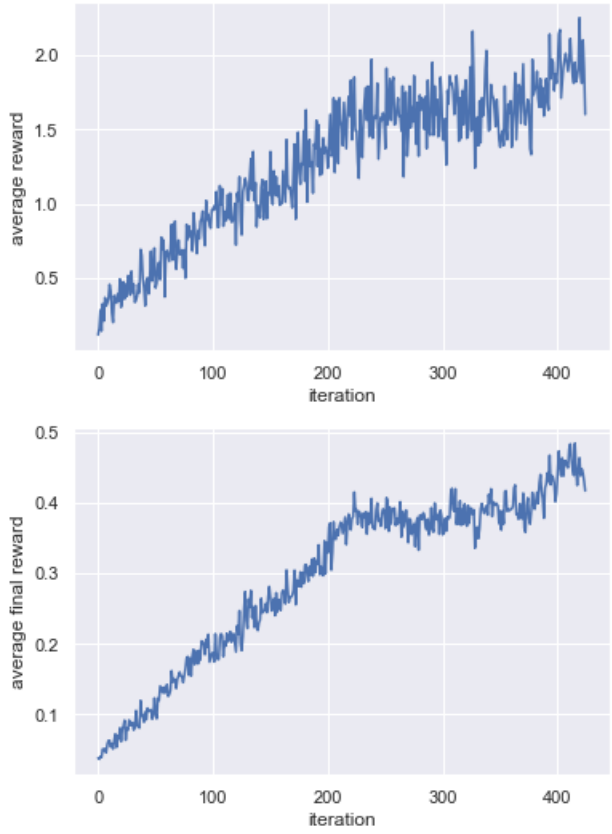
- Analysis 3: peptide length distribution





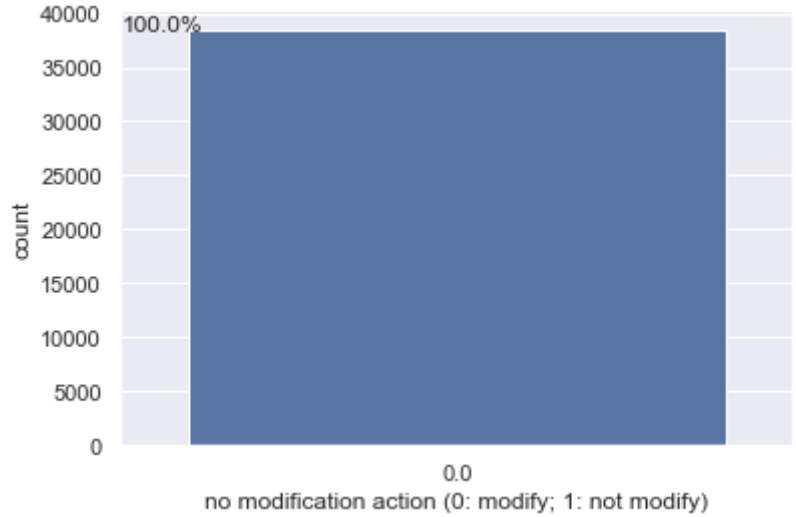
4. Result Analysis with different sampling strategy.

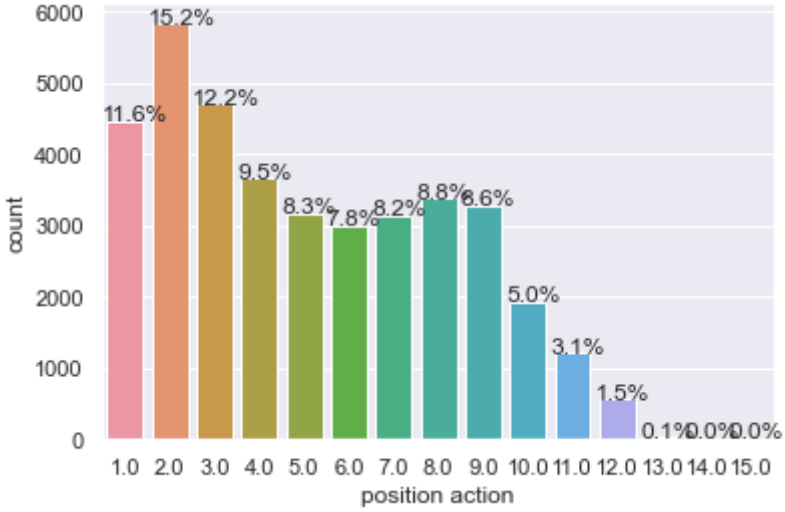
- Setting 2 with 50% IEDB samples and 50% randomly generated peptides

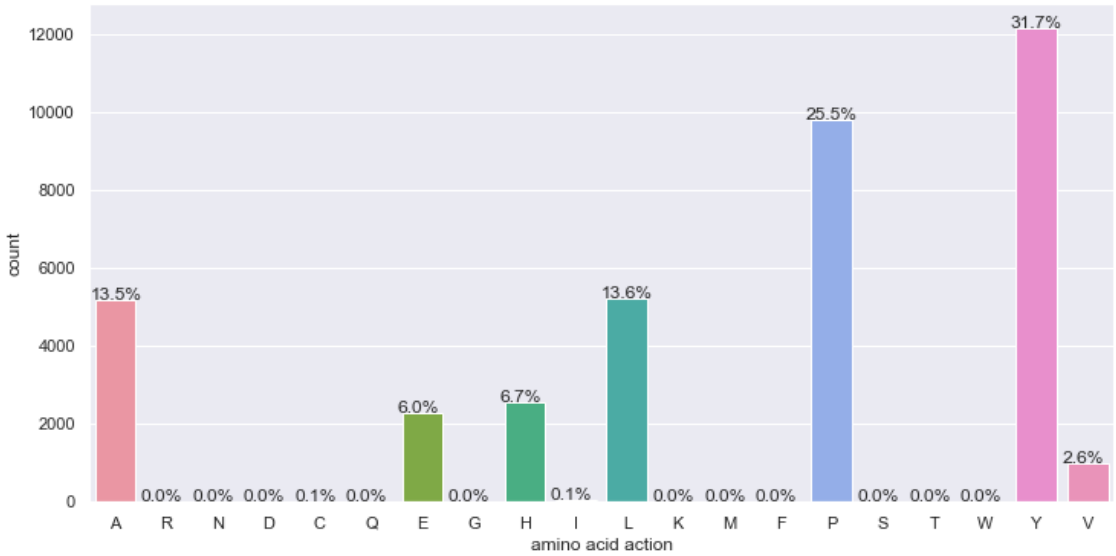


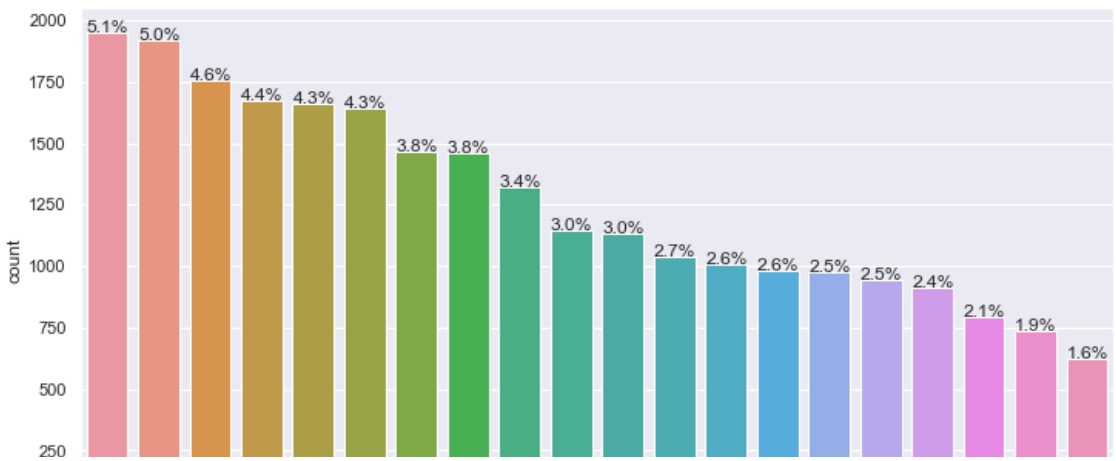
Each iteration has 3,840 steps. In each step, a specific given state will be updated by the policy network. Within 425 iterations, we have 204,000 samples in total. In order to analyze the trained policy network, we select the corresponding optimized peptides (i.e., the peptides in the terminal state) of the last 4,800 (about 2.4%) samples in the last 10 iterations for further analysis.

  - Analysis 1: action analysis

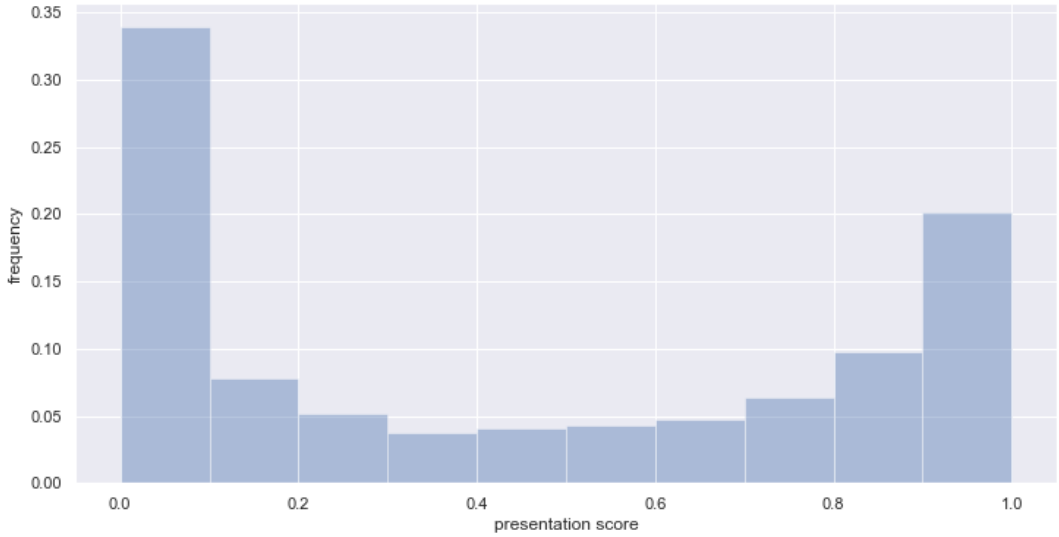

40000
35000
30000
25000
20000
15000
10000
5000
0
count
100.0%
0.0
no modification action (0: modify; 1: not modify)


6000
5000
4000
3000
2000
1000
0
count
11.6%
15.2%
12.2%
9.5%
8.3%
7.8%
8.2%
8.8%
8.6%
5.0%
3.1%
1.5%
0.1%
0.0%
0.0%
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0
position action

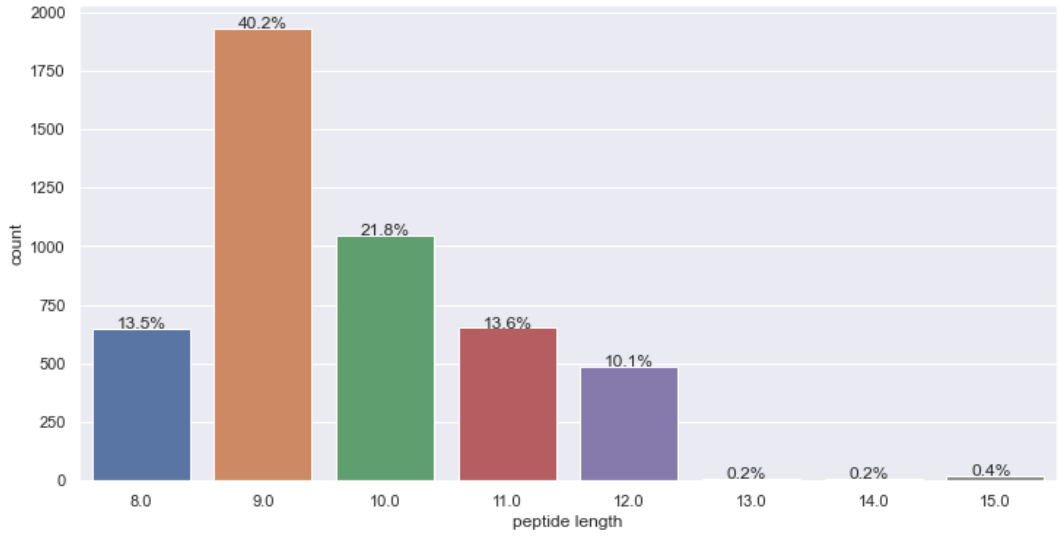

12000
10000
8000
6000
4000
2000
0
count
13.5%
0.0%
0.0%
0.0%
0.1%
0.0%
6.0%
0.0%
6.7%
0.1%
13.6%
0.0%
0.0%
0.0%
25.5%
0.0%
0.0%
0.0%
31.7%
2.6%
A R N D C Q E G H I L K M F P S T W Y V
amino acid action

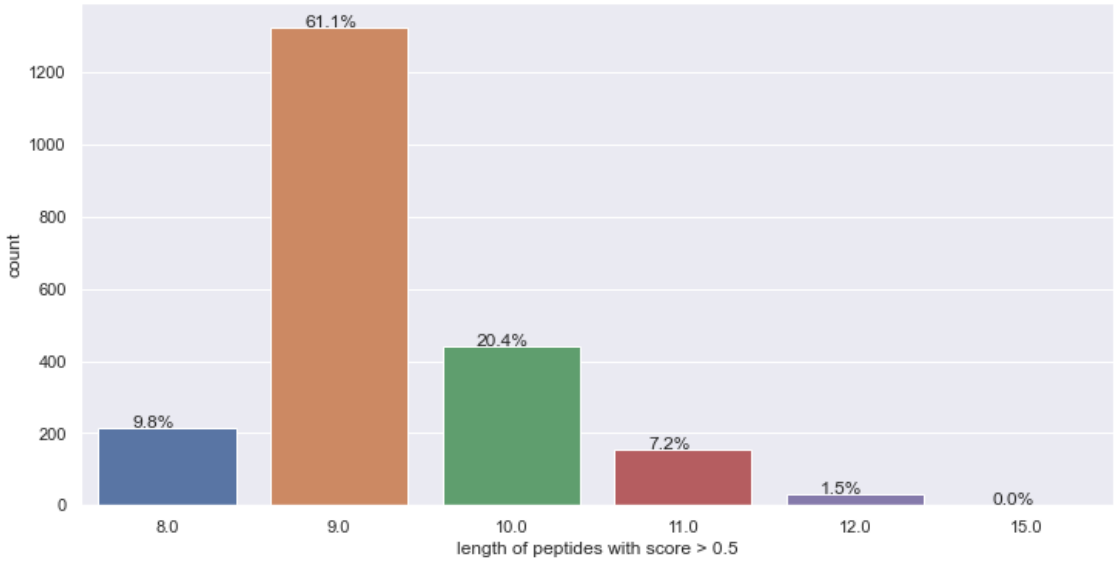

2000
1750
1500
1250
1000
750
500
250
count
5.1%
5.0%
4.6%
4.4%
4.3%
4.3%
3.8%
3.8%
3.4%
3.0%
3.0%
2.7%
2.6%
2.6%
2.5%
2.5%
2.4%
2.1%
1.9%
1.6%

- Analysis 2: Reward distribution (45.27% optimized peptides have the presentation scores > 0.5)

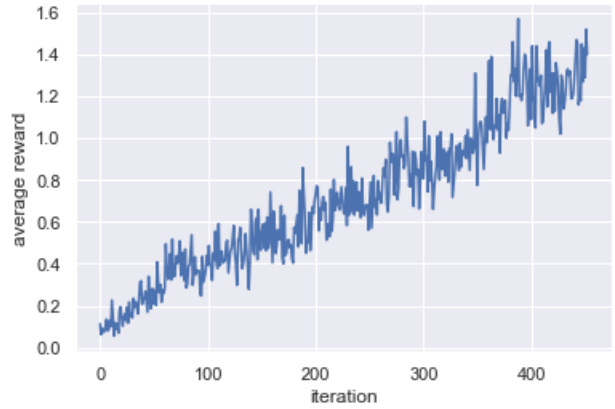


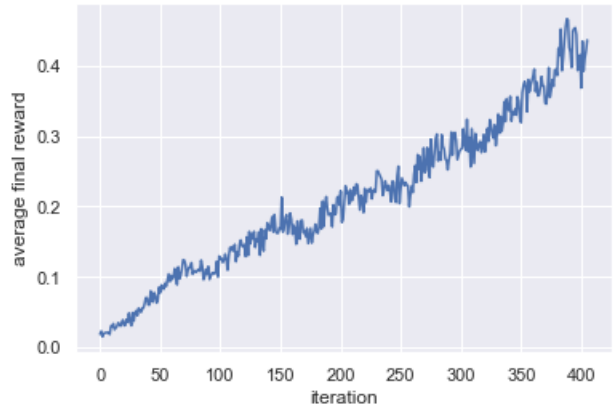
- Analysis 3: peptide length distribution





- Setting 2 with all randomly generated peptides





Update (5-19)

Suggestions:

1) Using a large entropy coefficient at the beginning to encourage exploration and then decrease the entropy coefficient. To avoid the learned policies being conserved at certain amino acids, it can be helpful to improve the exploration using a large coefficient.

2) Using prioritized experience replay (i.e., importance sampling) to increase the exploitation of good episodes (i.e., episodes with good presentation scores and limited modification).

3) For constrained generation (i.e., generating positive peptides from Covid-19 peptides), comparing the performance of the model with prioritized experience replay and the model with less maximum steps (change maximum number of steps to 3 or 4).

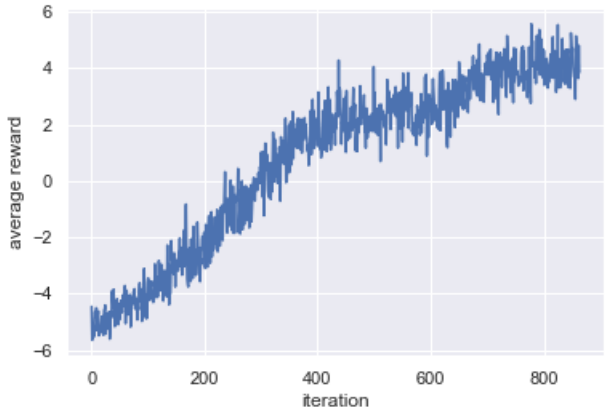
4) Improving the performance of models under the setting with only final rewards, through warming up (i.e., using the intermediate rewards at the beginning of training and then removing the intermediate rewards) or using the prioritized replay. In addition, trying to change the "no modification" action to "termination" action so that shorter episodes with large representation scores can receive higher cumulative rewards than longer episodes. Note that "termination" action can lead to less exploration and thus worse performance. Therefore, we need to limit the usage of "termination" action at the beginning of training.

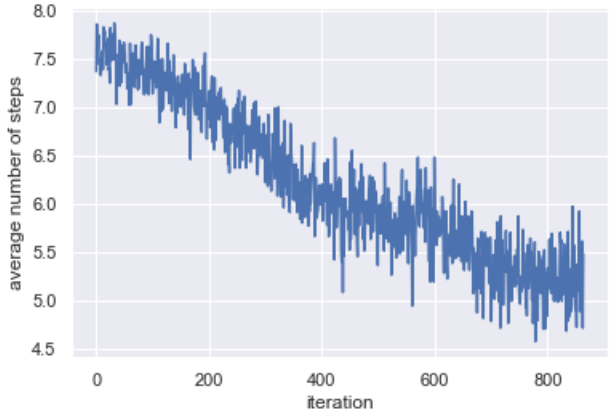
5)

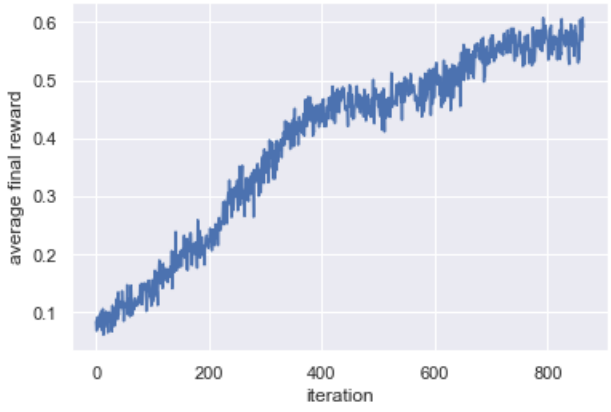
1. Result Analysis with longer running time

- Setting 1 (+1, -1, -0.9) (stop criteria = 0.6; sample rate = 80% from IEDB and 20% from random)

  It takes 72 hours to get the results below....



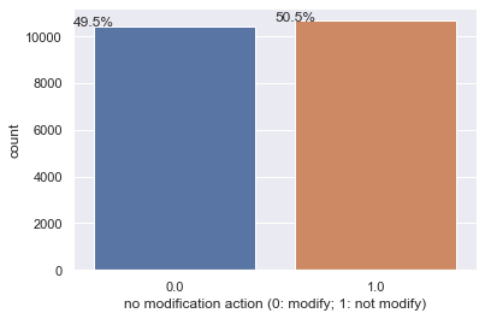


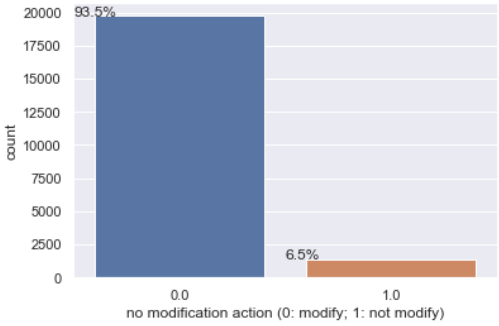


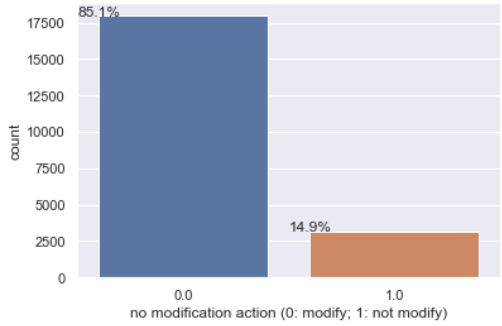
Action analysis

1. no modification action

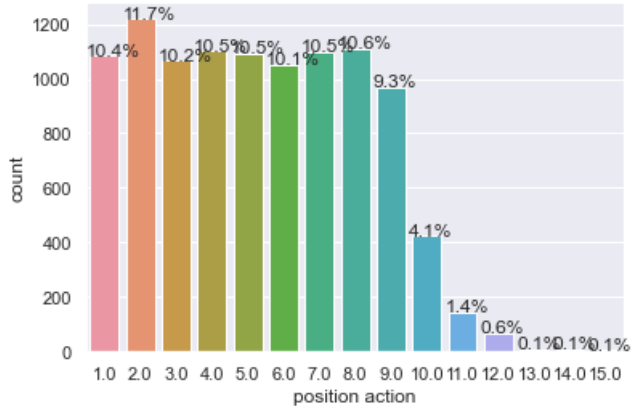
   (first 10 iterations) (mid 10 iterations) (last 10 iterations)

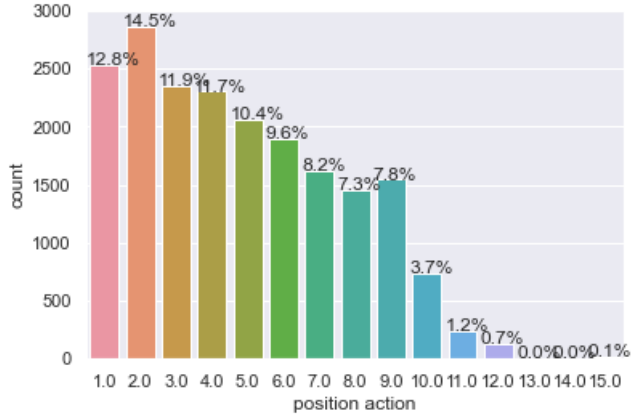


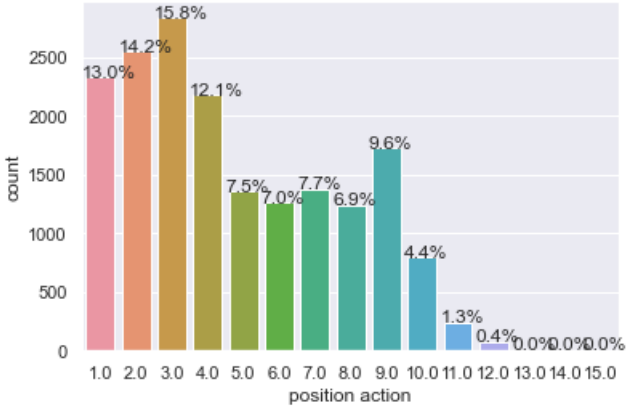




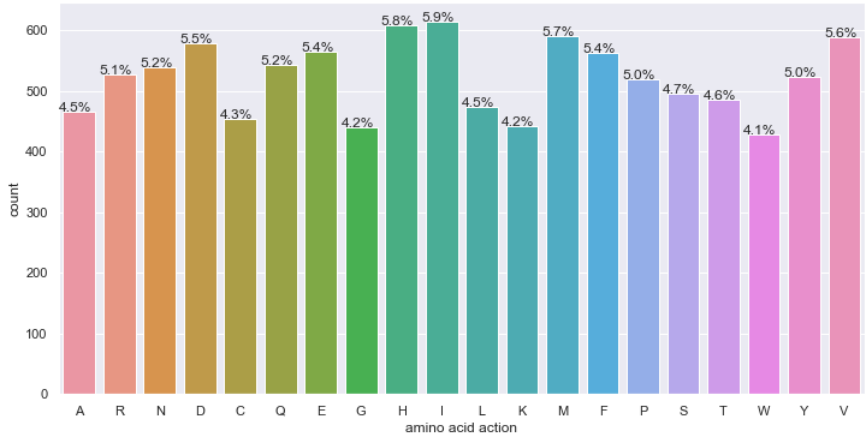
2. position select action

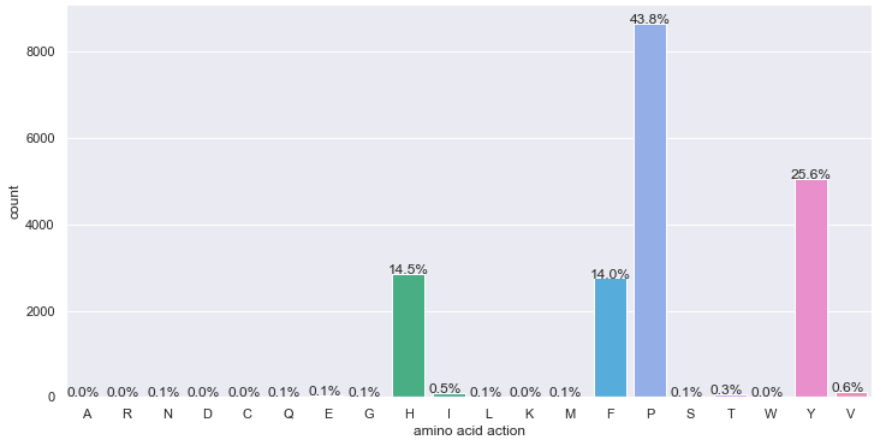


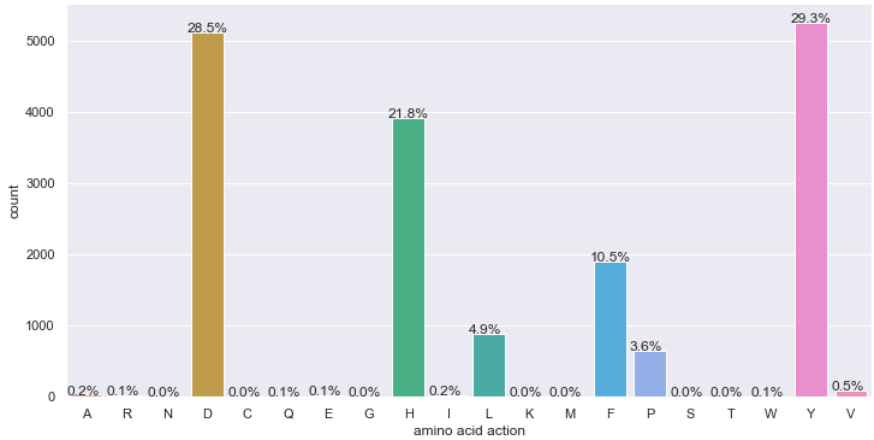




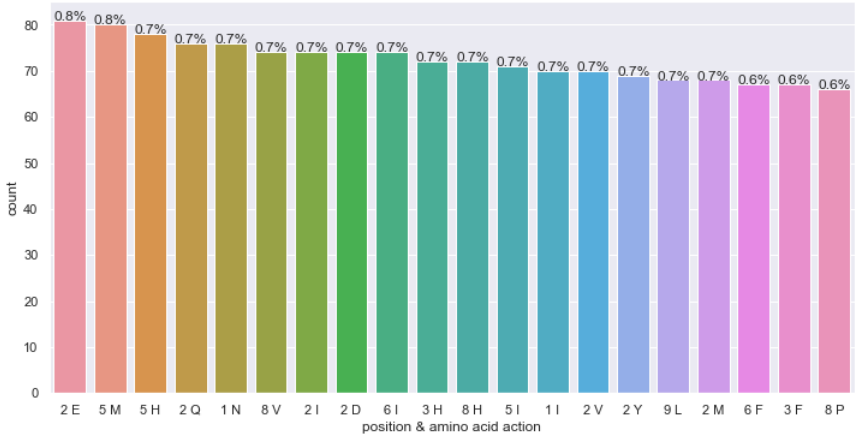
3. amino acid action

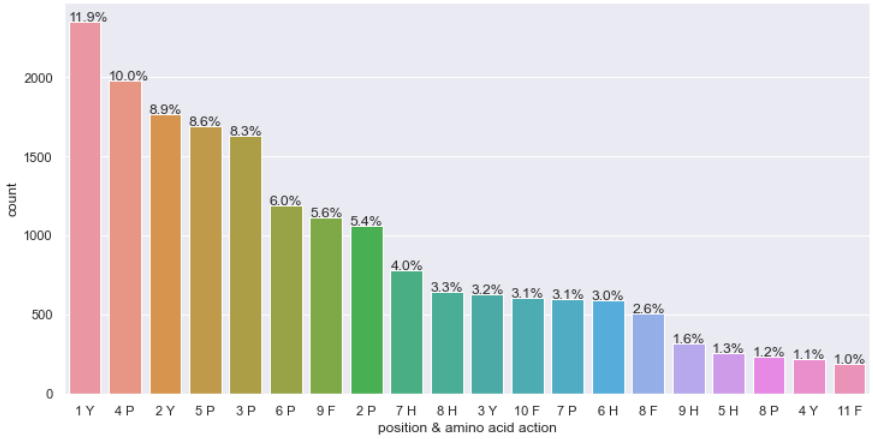


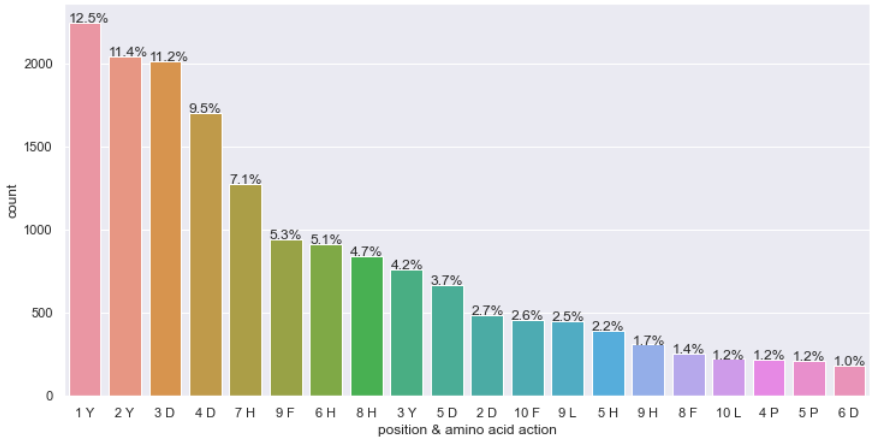




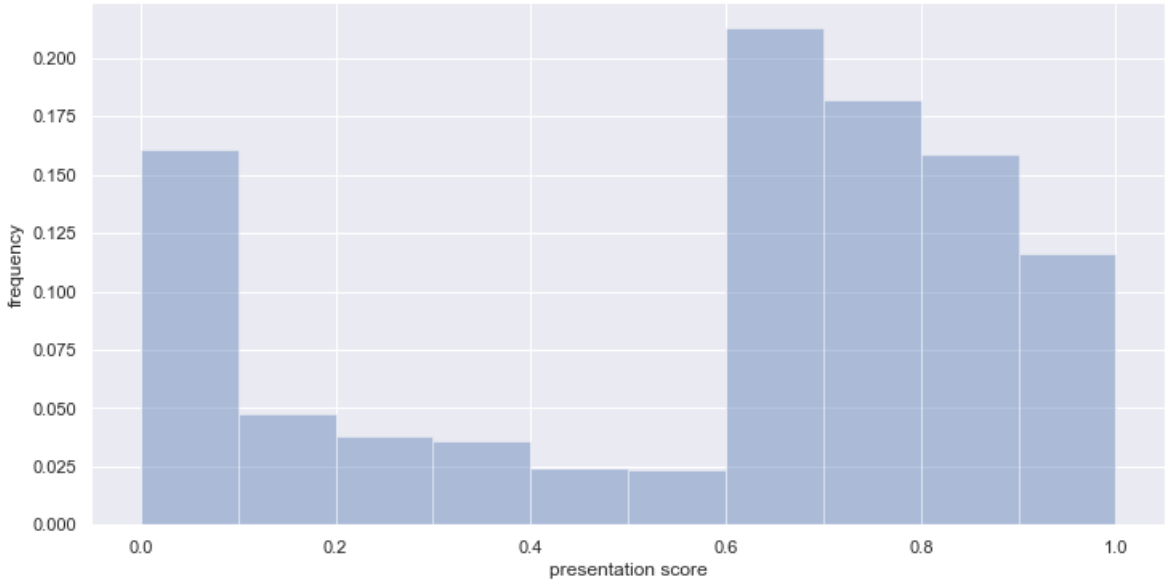
4. pos & amino action



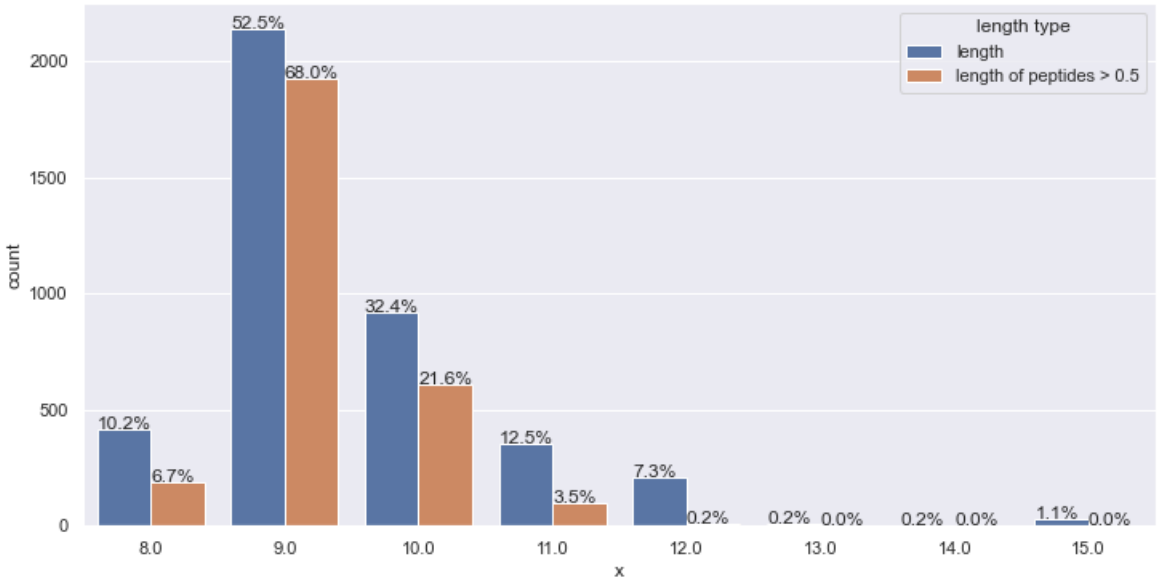




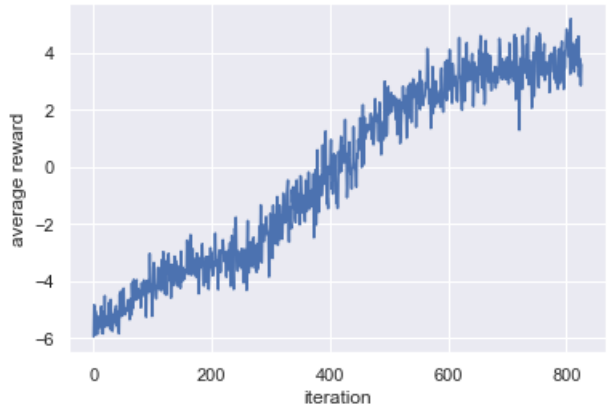
Presentation score analysis: (69.4% optimized peptides have presentation scores > 0.5)

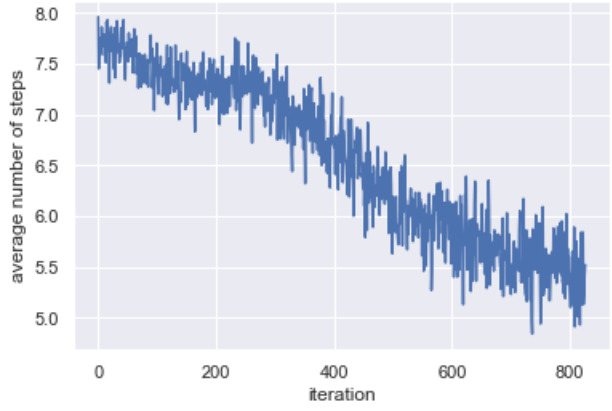


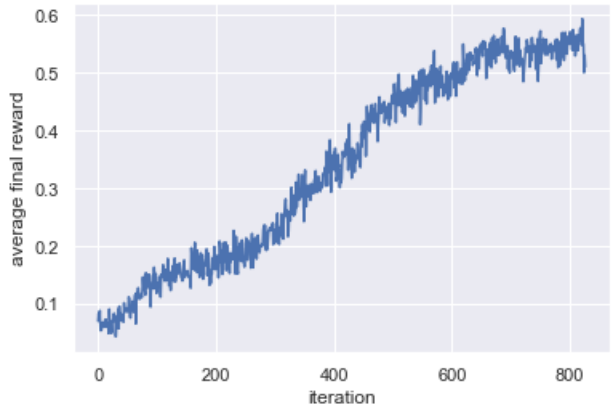
Peptide length distribution



- Setting 1 (+1, -1, -0.9) (stop criteria = 0.6; sample rate = 50% from IEDB and 50% from random)

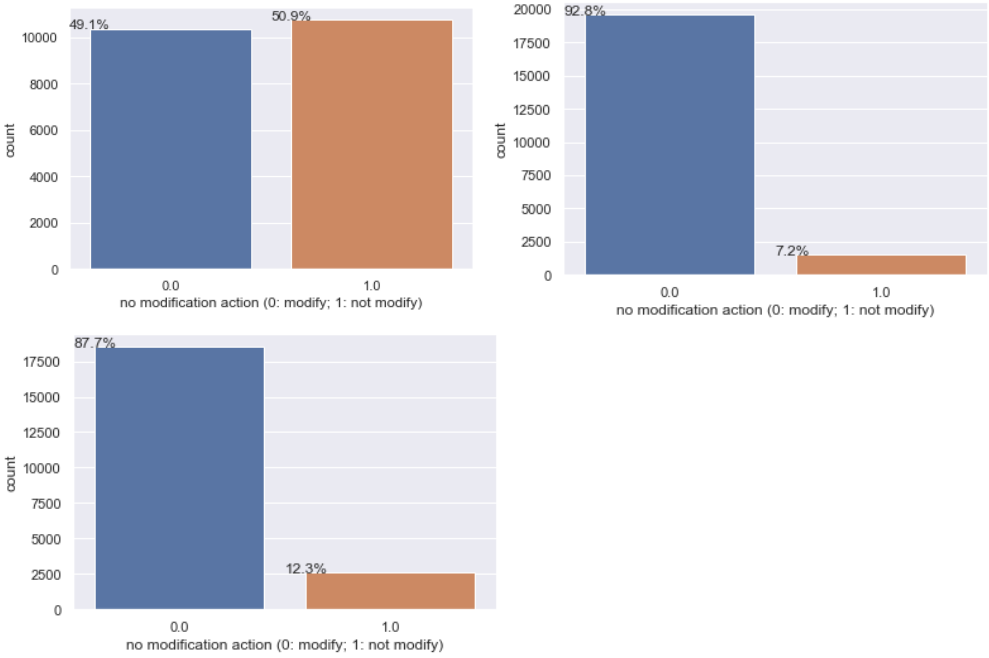




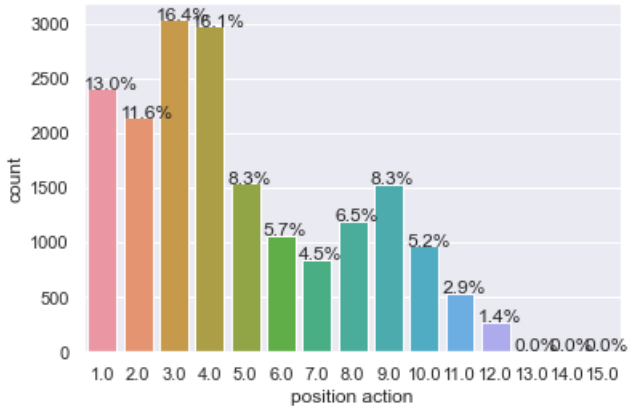


1. no modification action

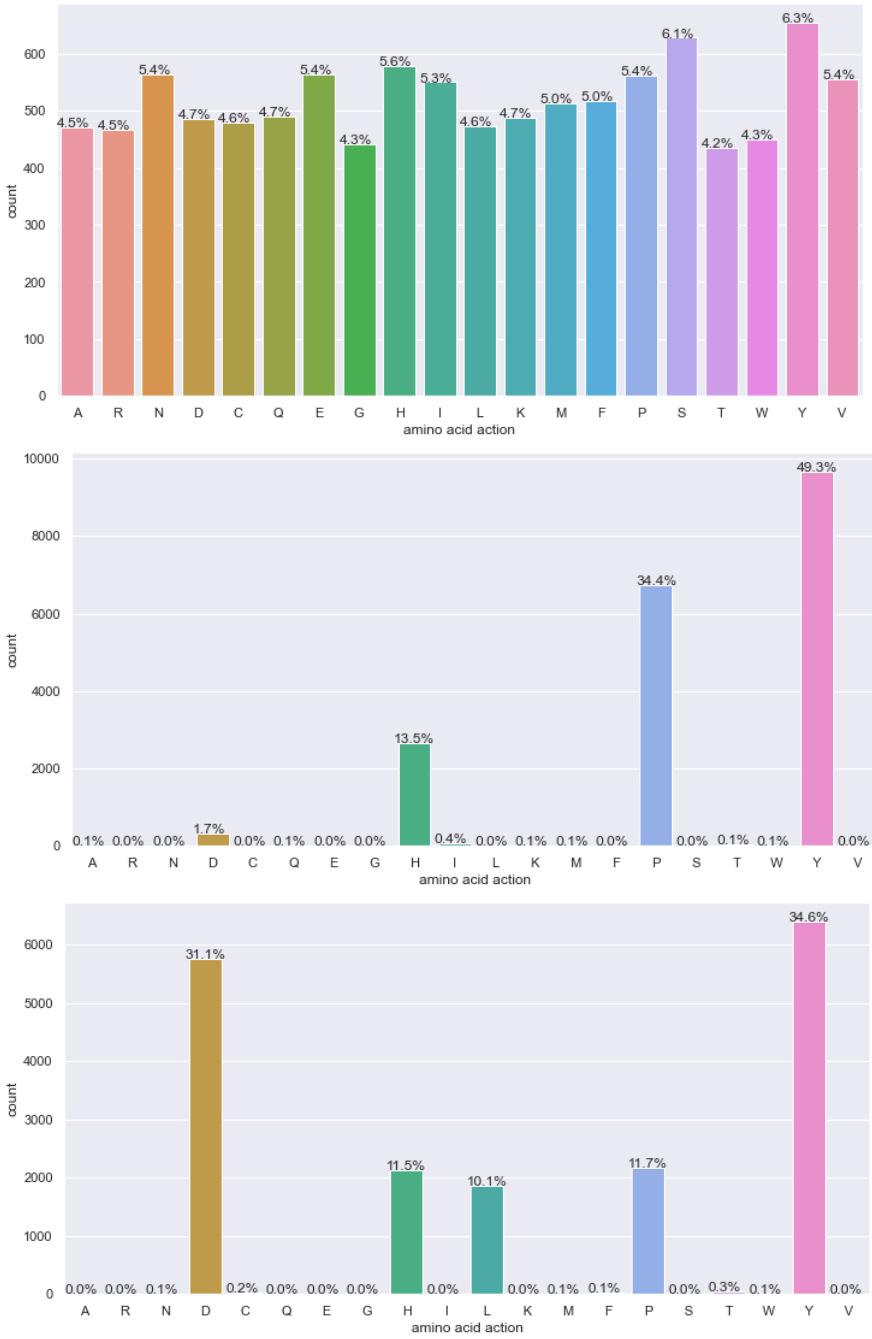
(first 10 iterations) (mid 10 iterations) (last 10 iterations)



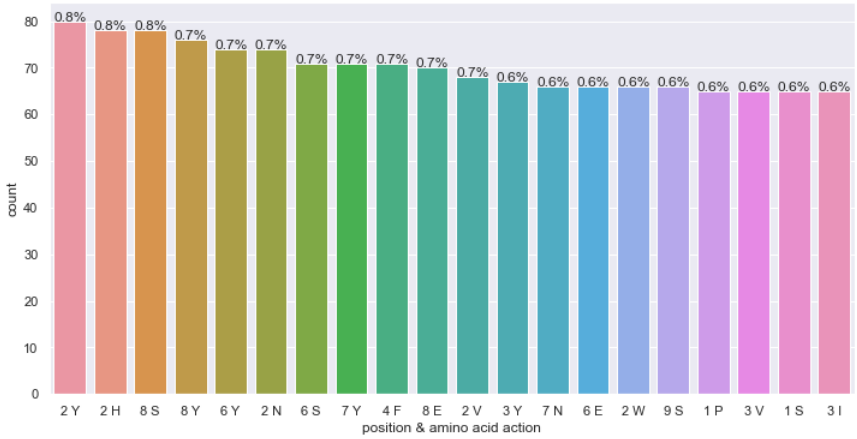
2. position select action

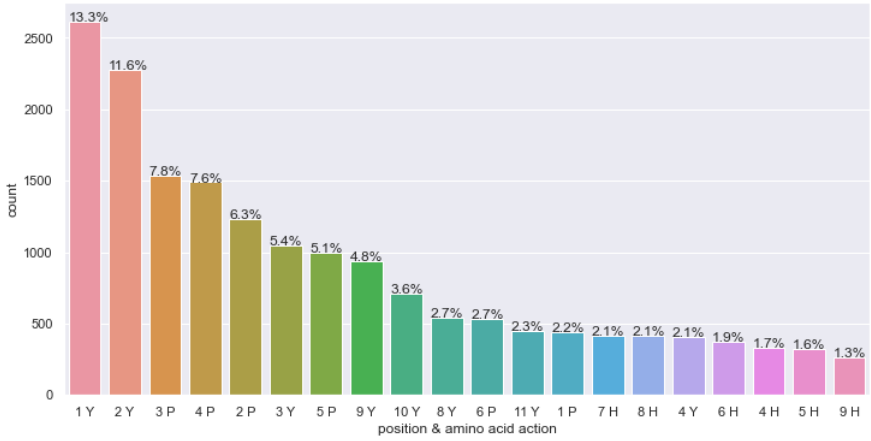


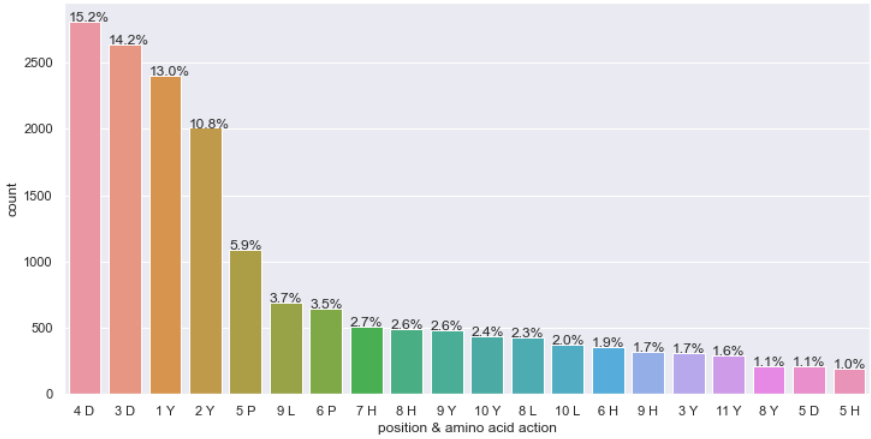
3. amino acid action



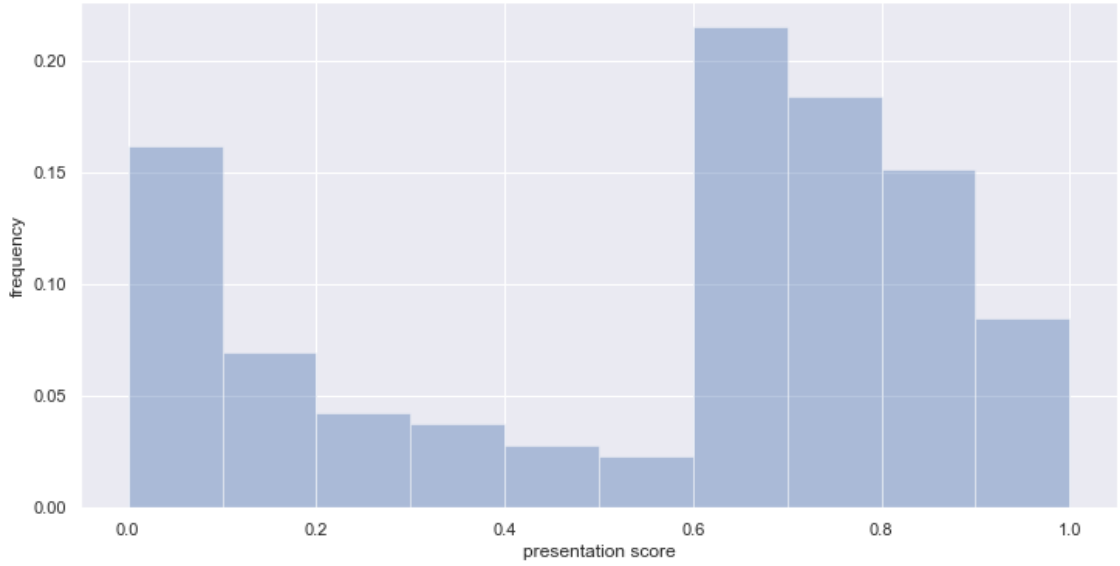
4. position & amino action

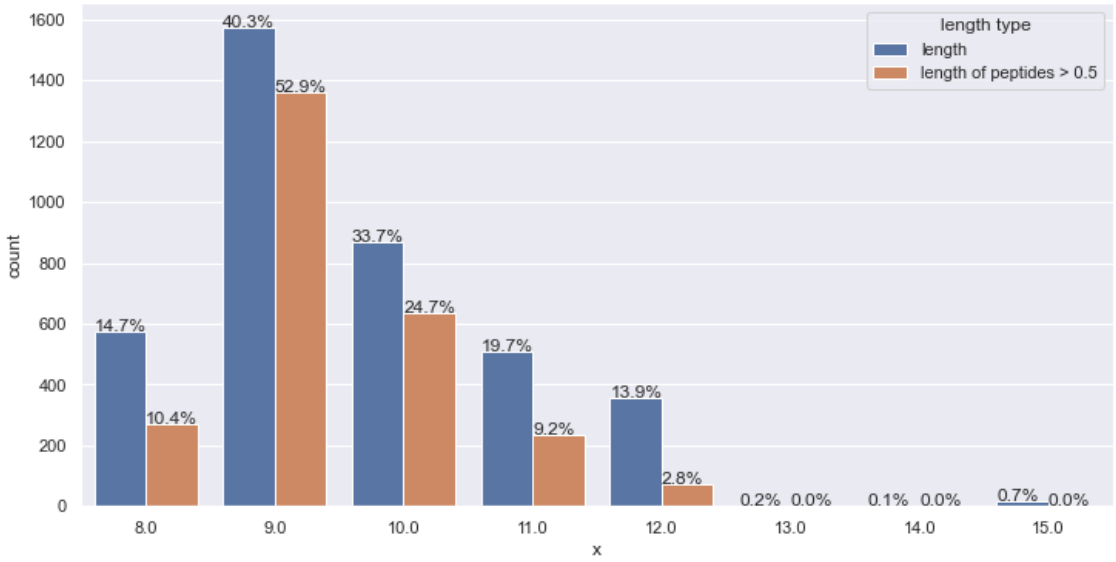




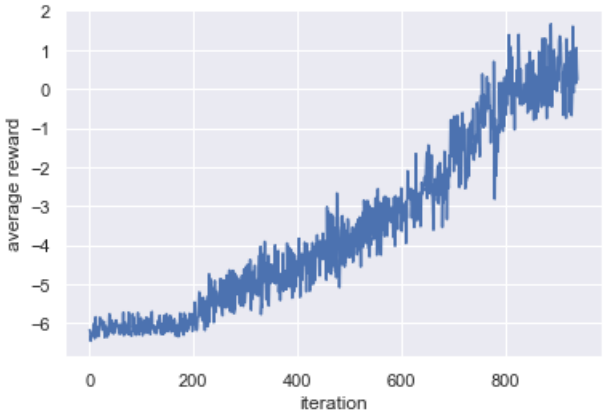


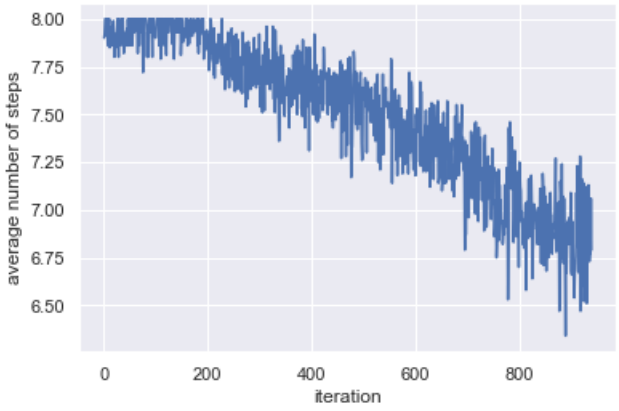
presentation score analysis: (65.9% optimized peptides have presentation scores > 0.5)

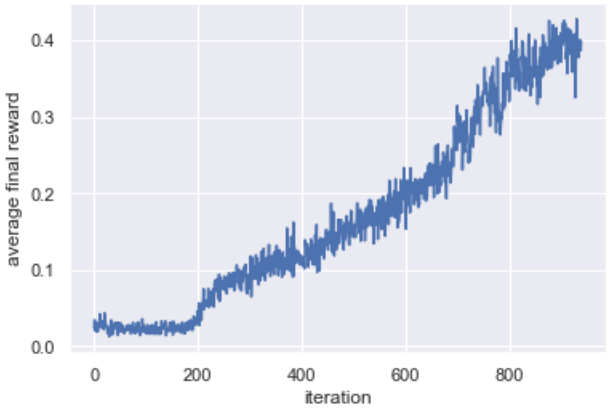




- Setting 1 (+1, -1, -0.9) (stop criteria = 0.6; sample rate = 100% from random)

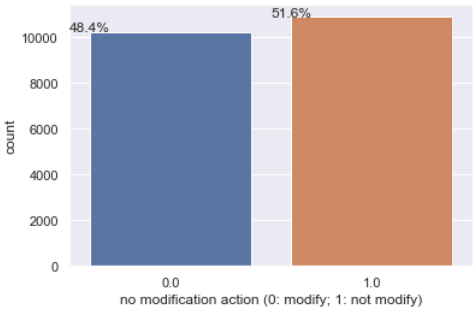


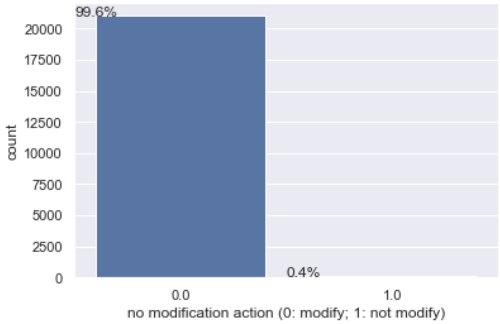


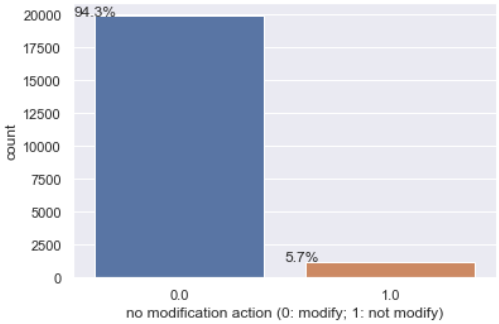


1. no modification action

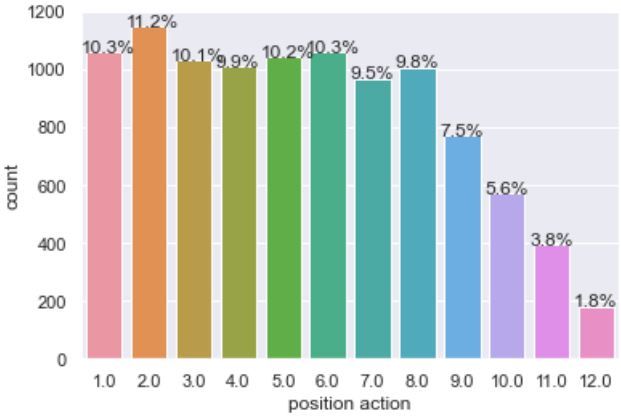
(first 10 iterations) (mid 10 iterations) (last 10 iterations)

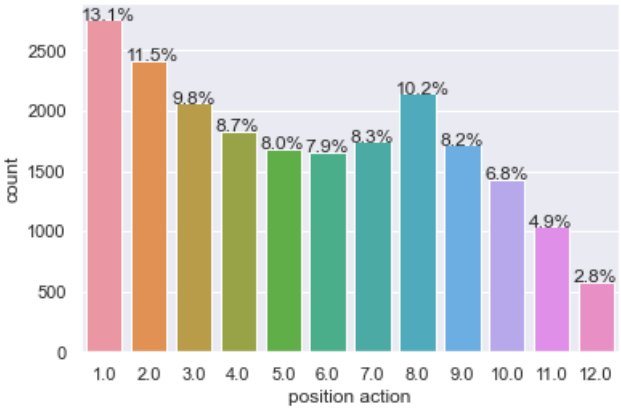


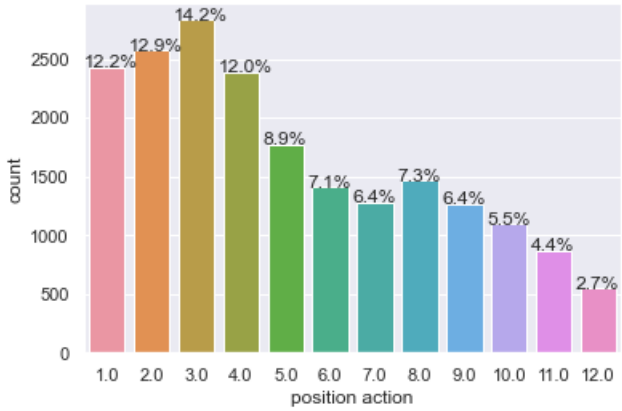




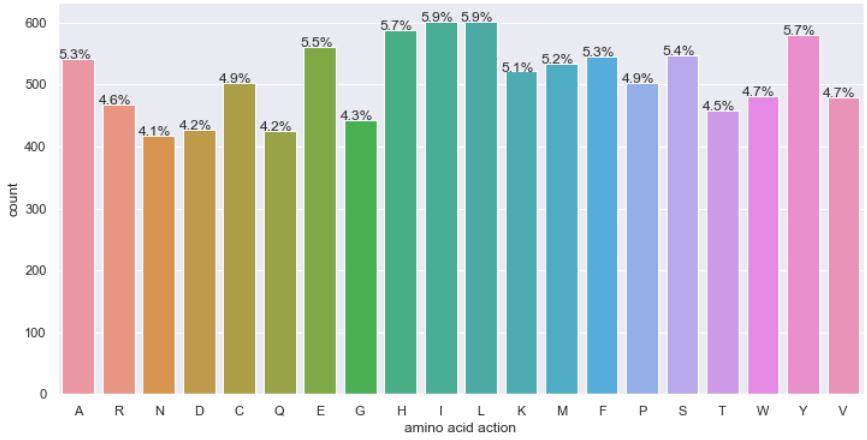
2. position select action

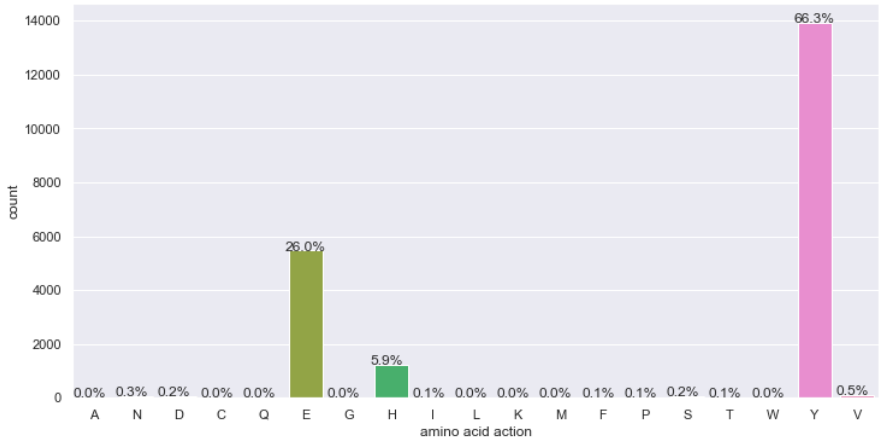


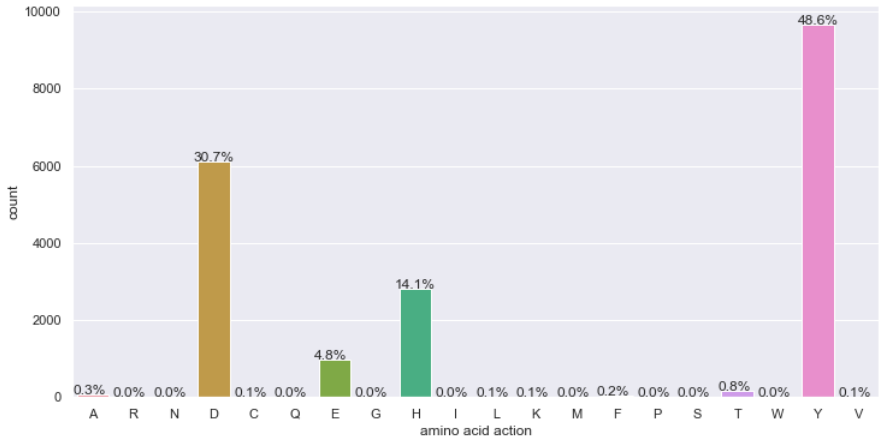




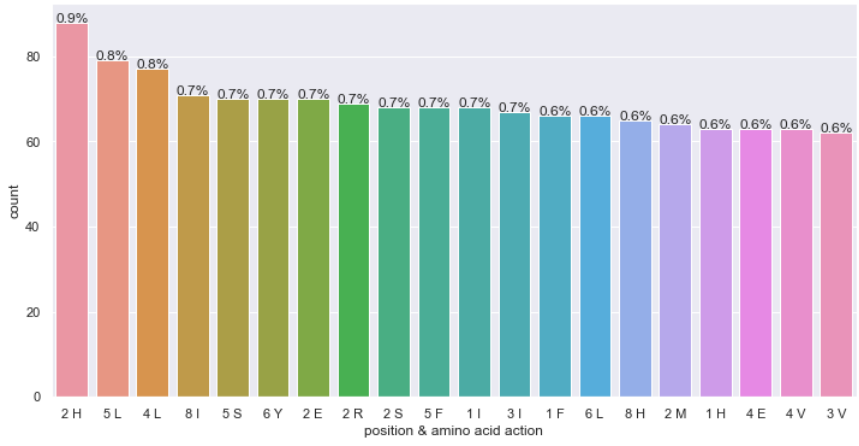
3. amino acid action

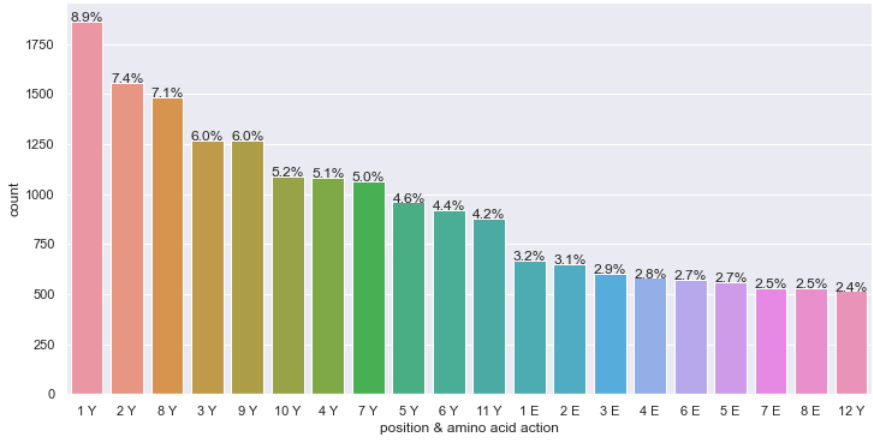


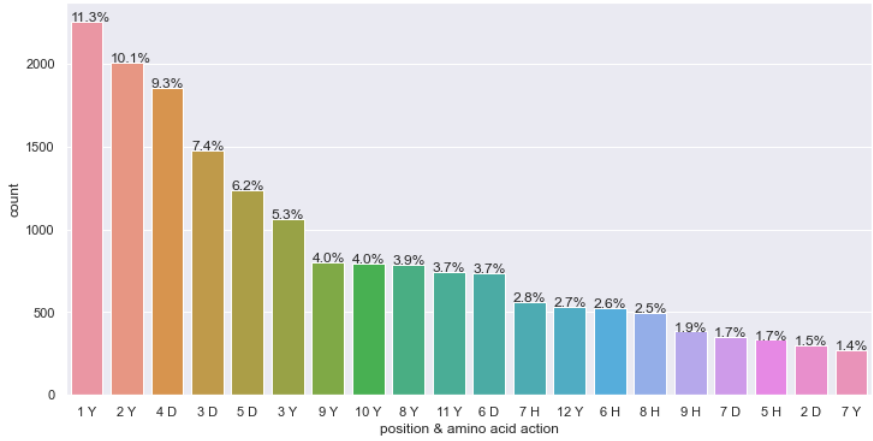




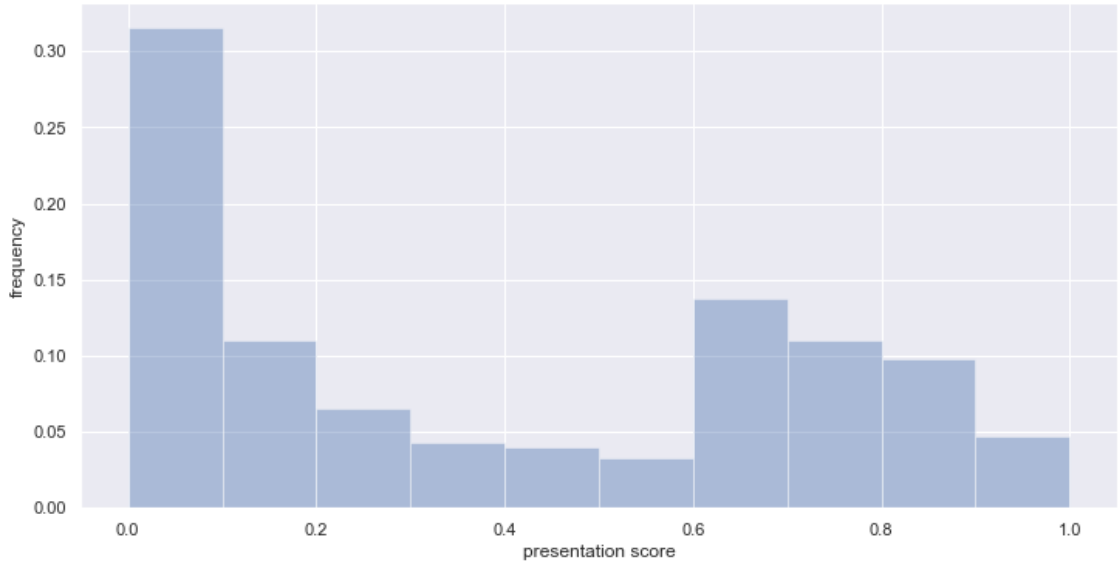
4. pos & amino action

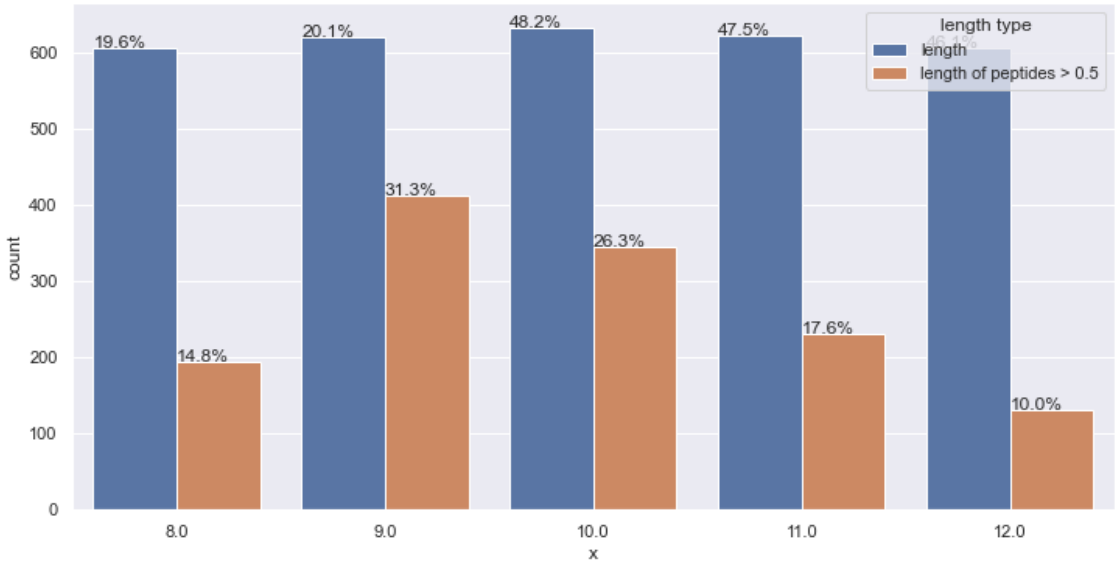






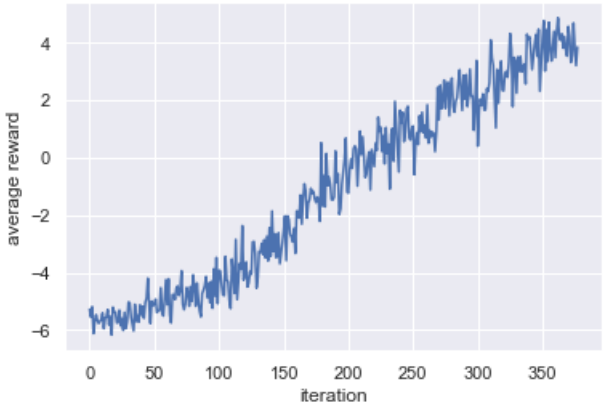
presentation score analysis: (42.5% optimized peptides have presentation scores > 0.5)

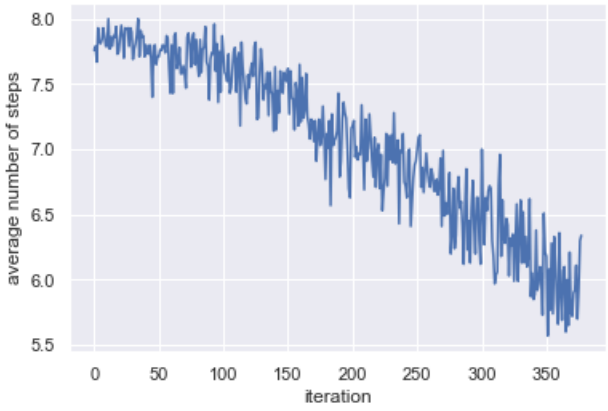


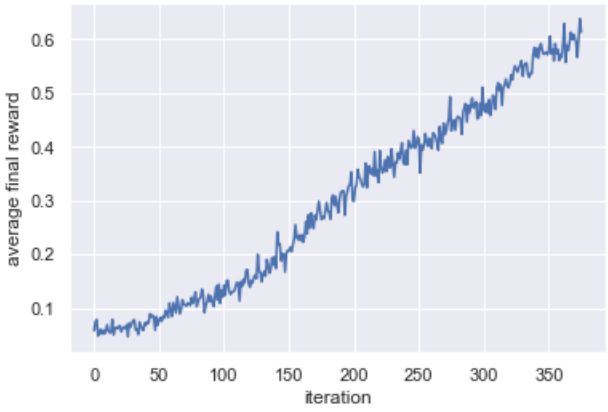


2. Result Analysis with different stopping criteria

- Setting 1 (+1, -1, -0.9) (stop criteria = 0.75; sample rate = 50% from IEDB and 50% from random)

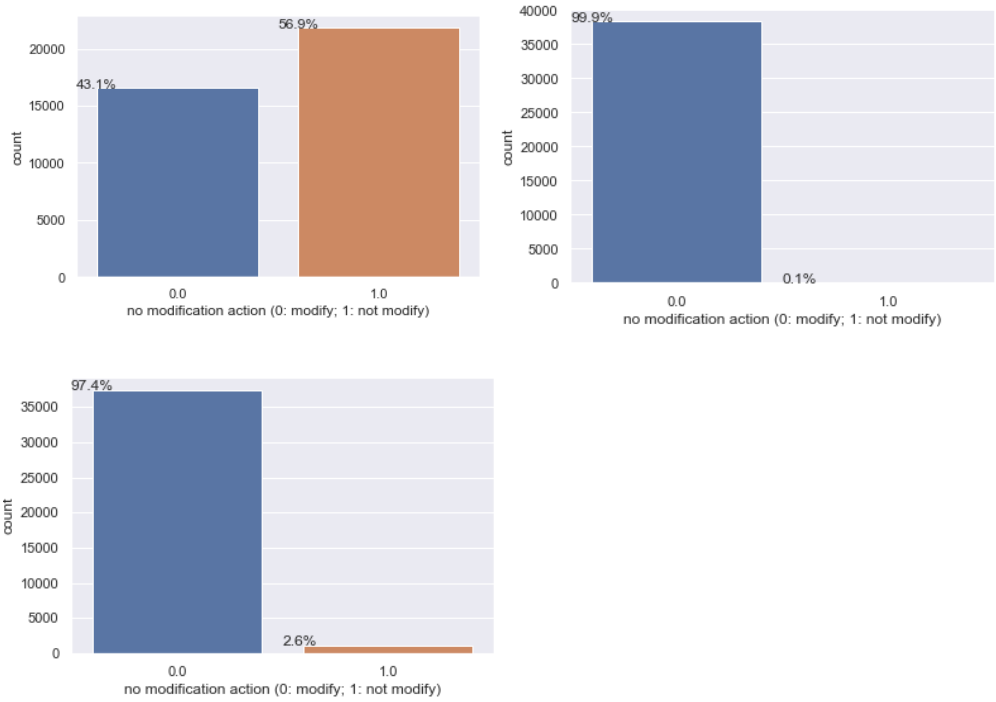




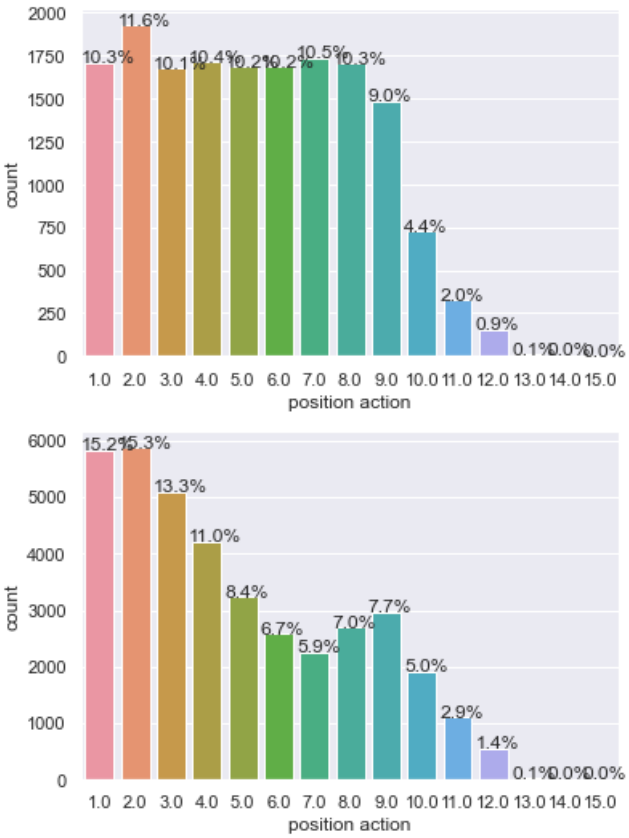


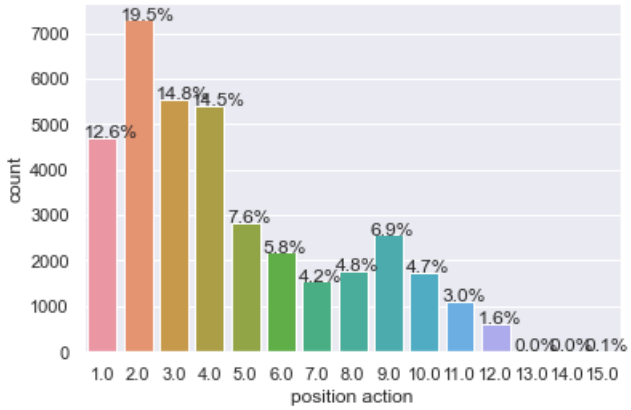
1. no modification action

(first 10 iterations) (mid 10 iterations) (last 10 iterations)



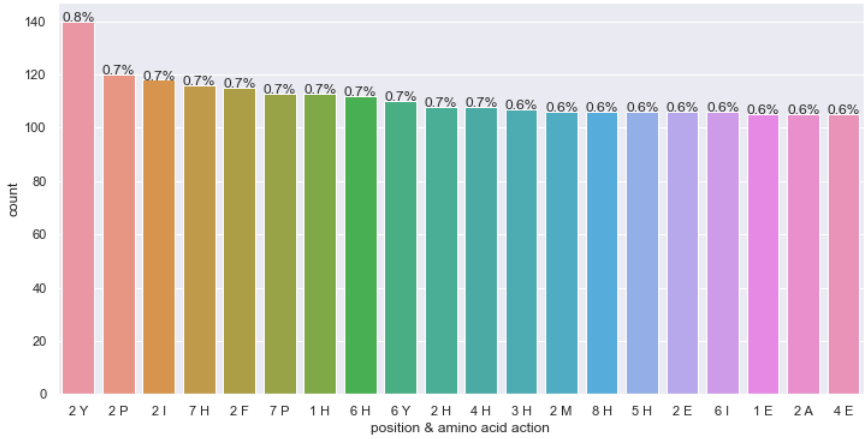
2. position select action

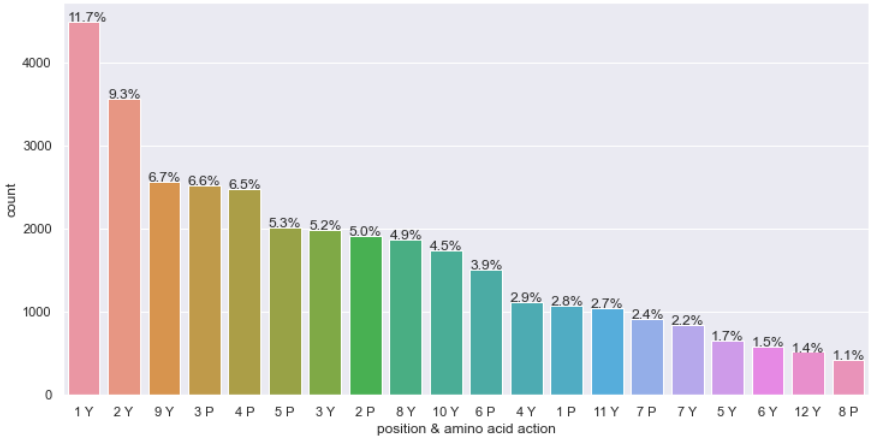


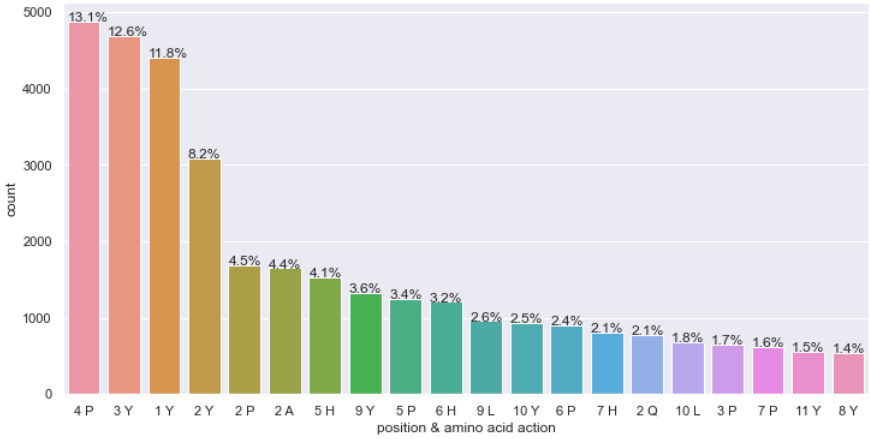


3. amino acid action

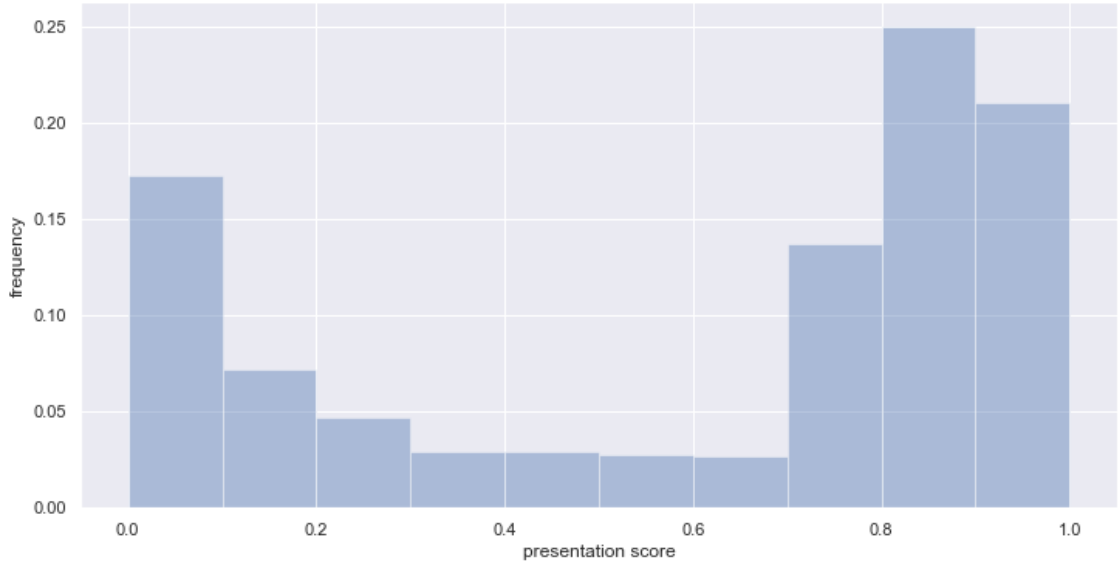
4. pos & amino action

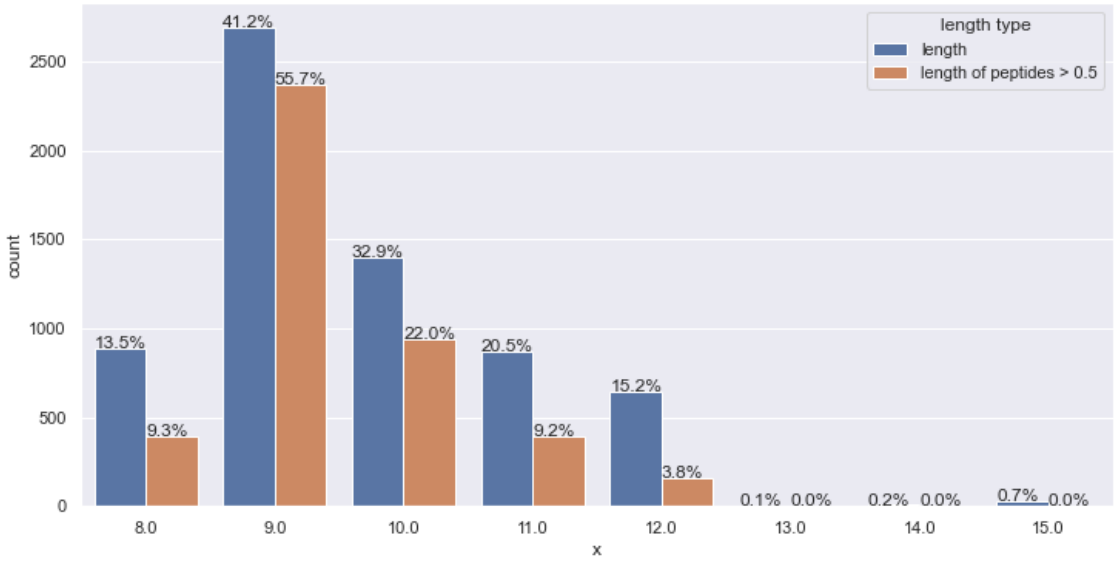




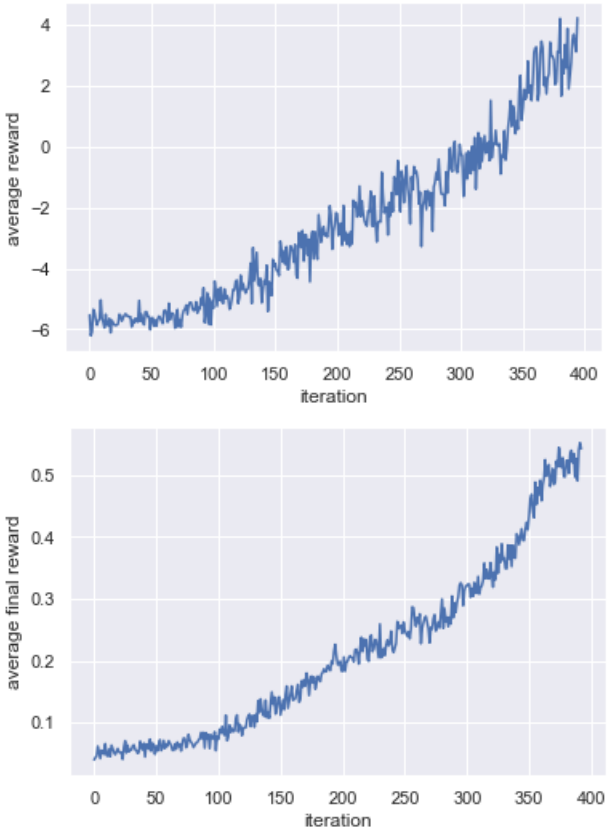


presentation score analysis: (65.1% optimized peptides have presentation scores > 0.5)

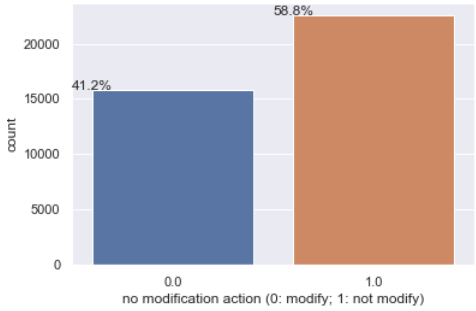


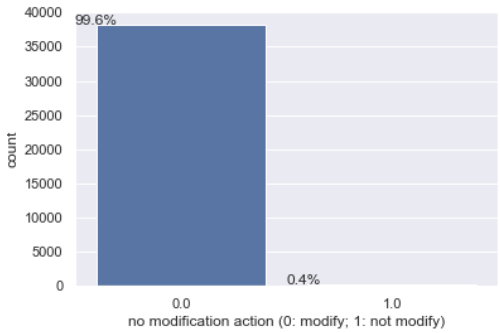


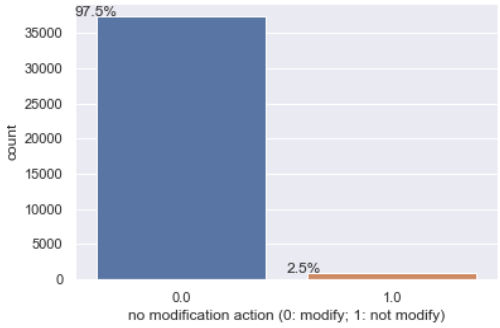
- Setting 1 (+1, -1, -0.9) (stop criteria = 1.0; sample rate = 50% from IEDB and 50% from random)



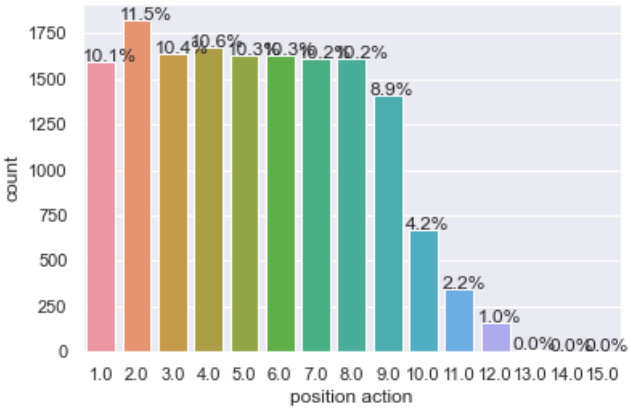
1. no modification action

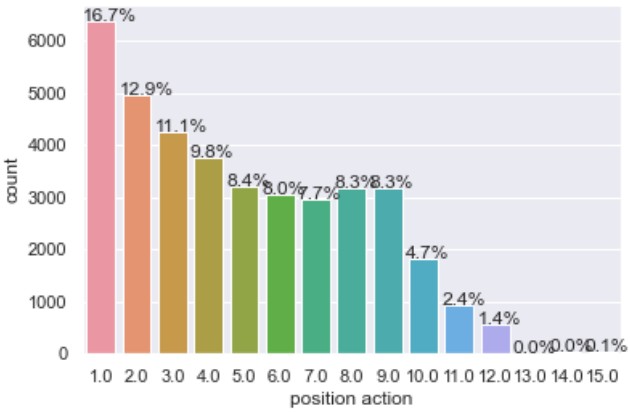


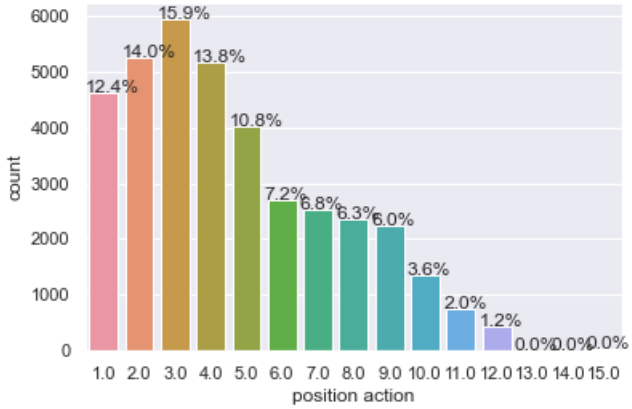




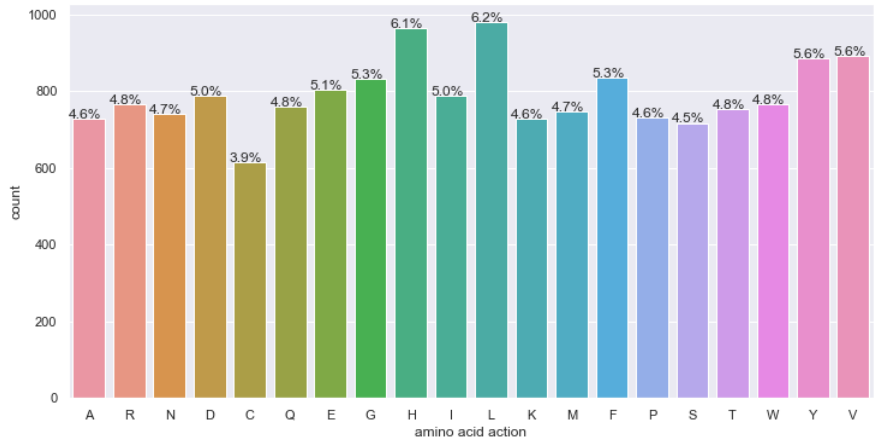
2. position select action

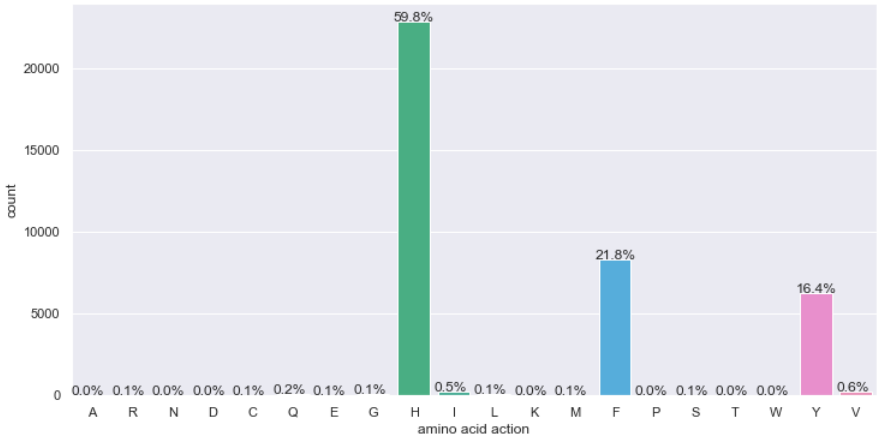


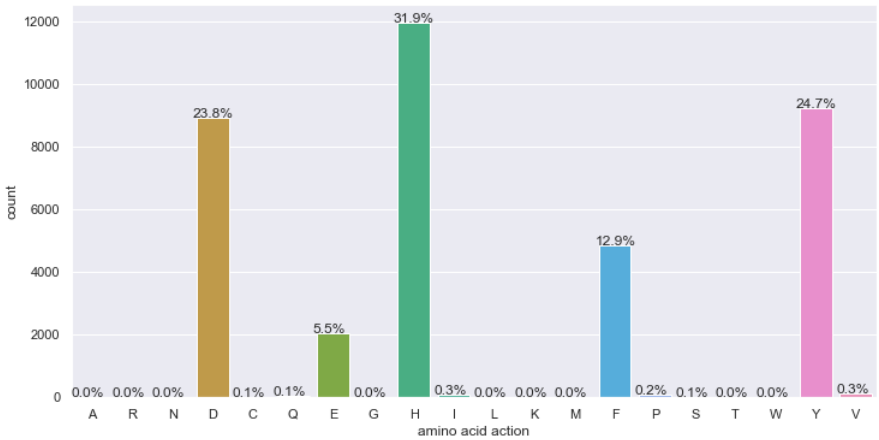




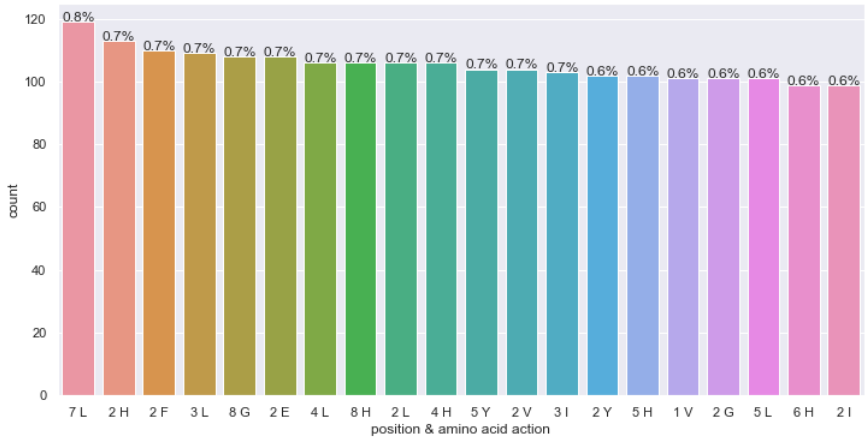
3. amino acid action

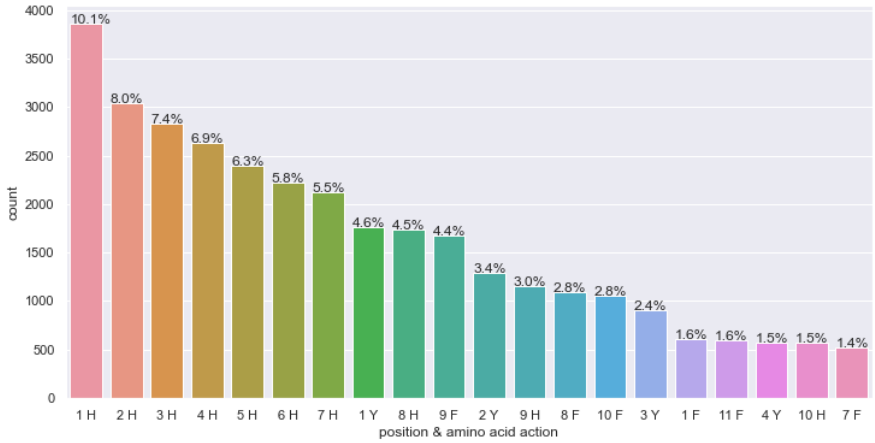


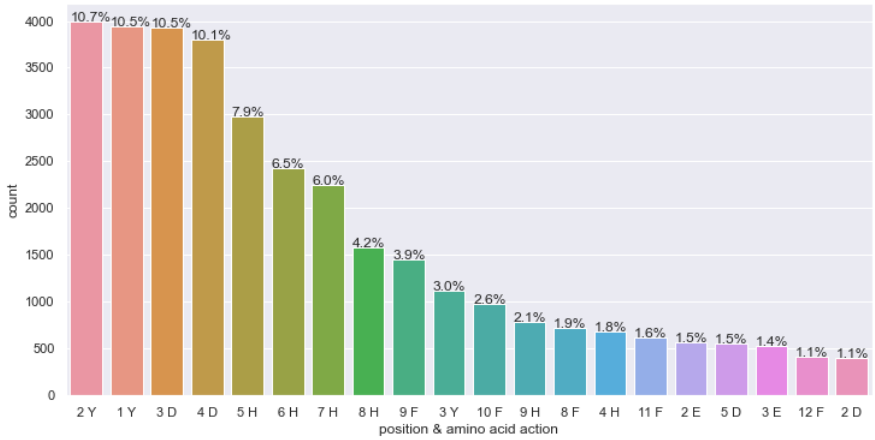




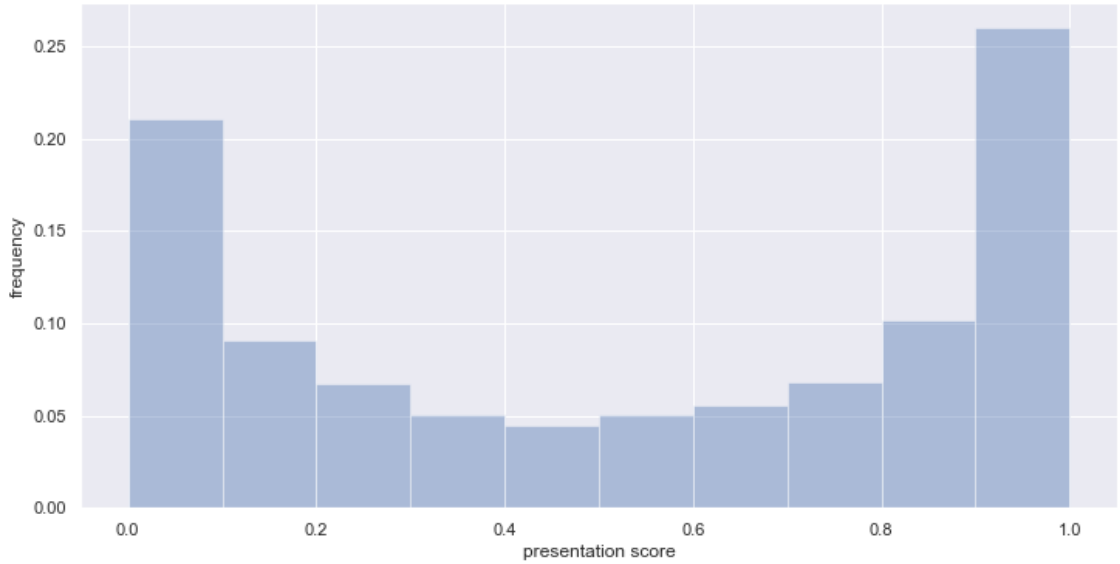
4. pos & amino action

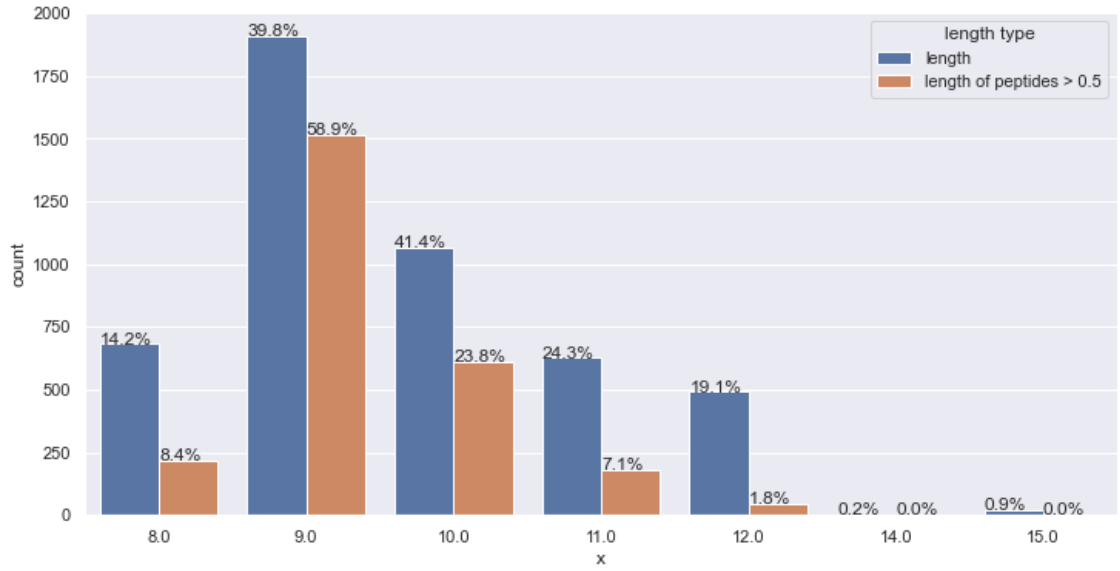






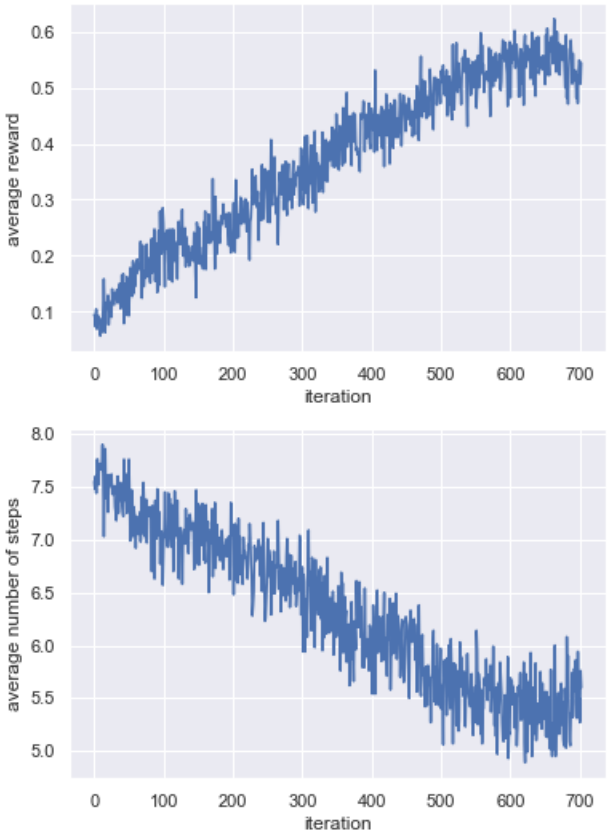
presentation score analysis: (53.6% optimized peptides have presentation scores > 0.5)



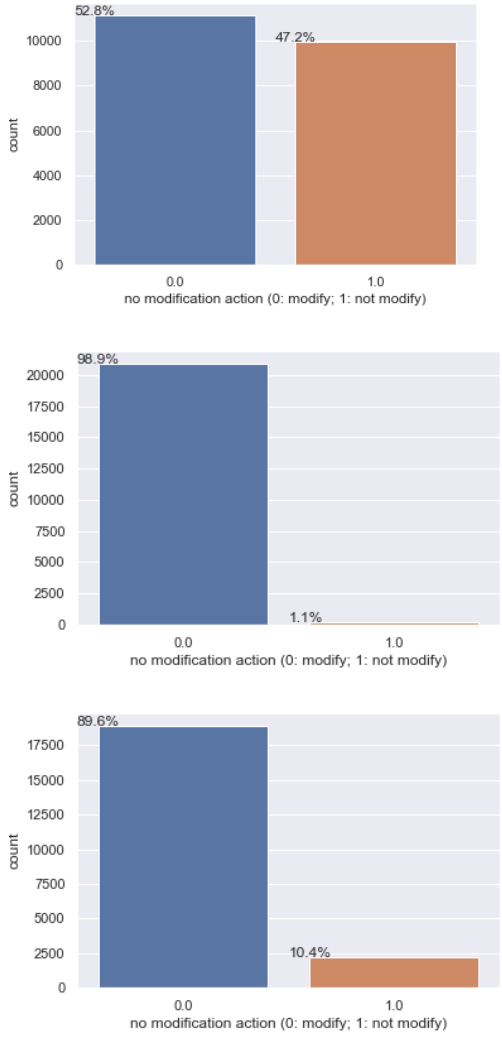


3. Result Analysis with only final rewards

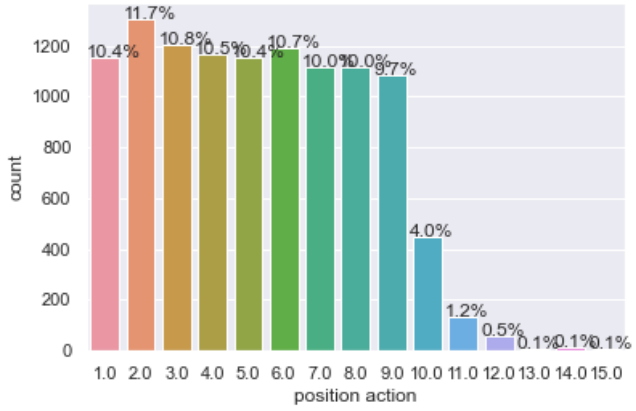
- Setting 1 (stop criteria = 0.6; sample rate = 80% from IEDB and 20% from random)

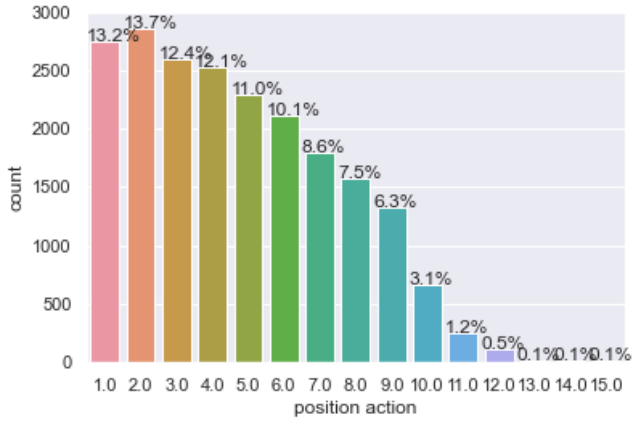


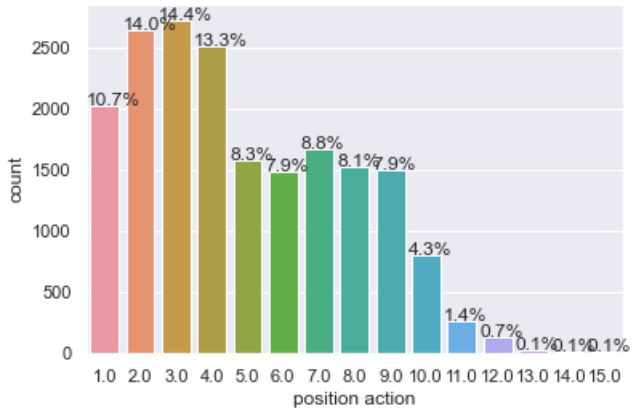
1. no modification action



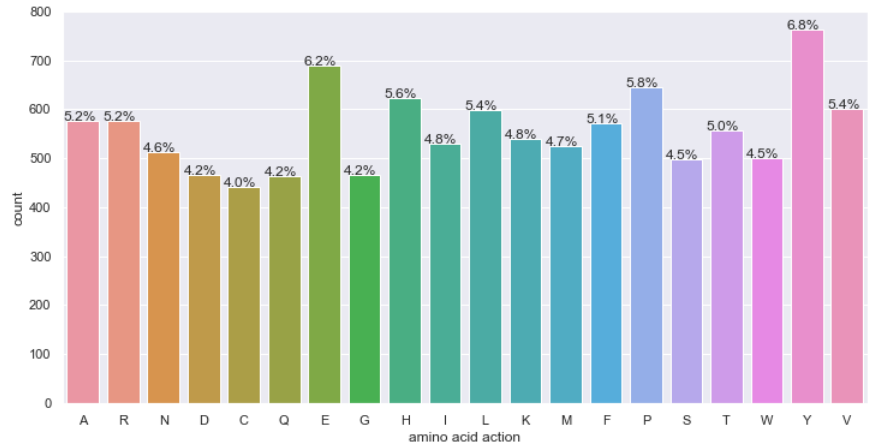
2. position select action

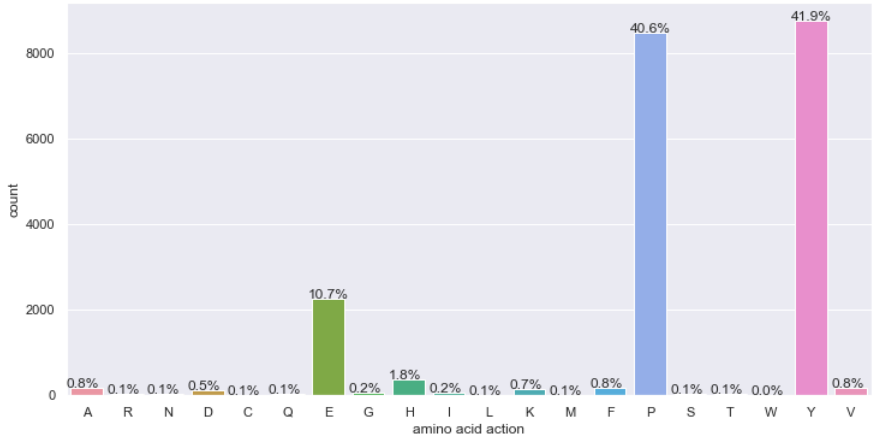


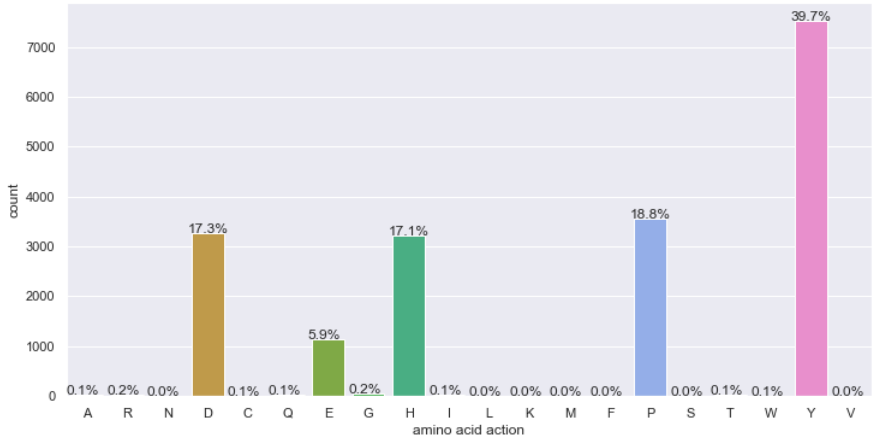




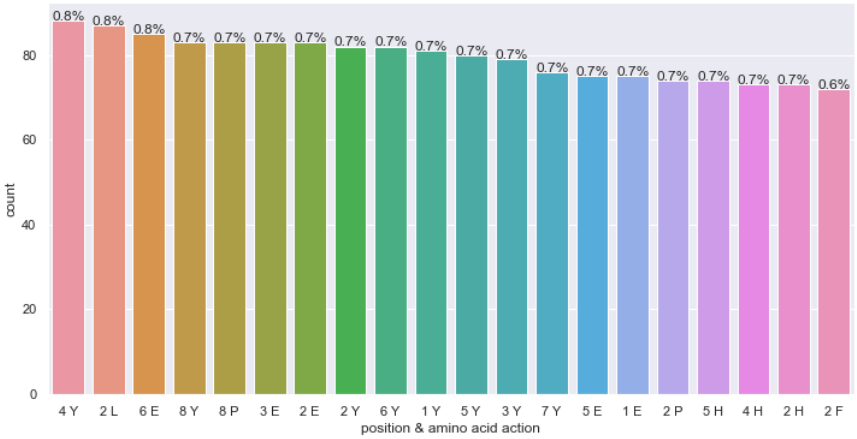
3. amino acid action

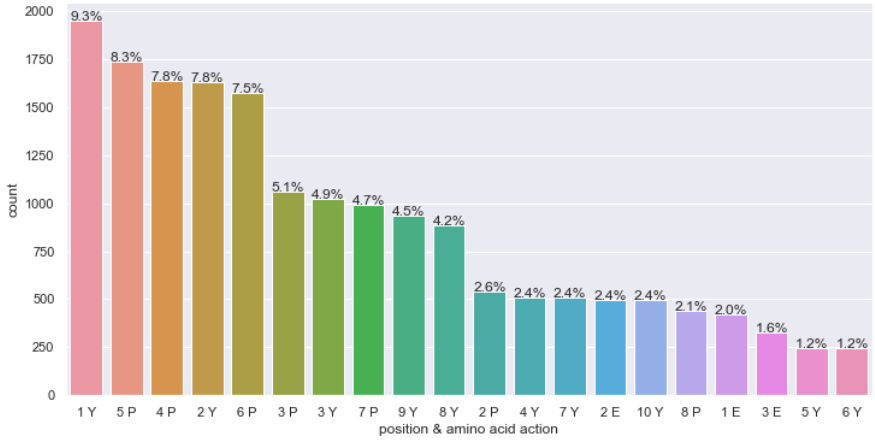


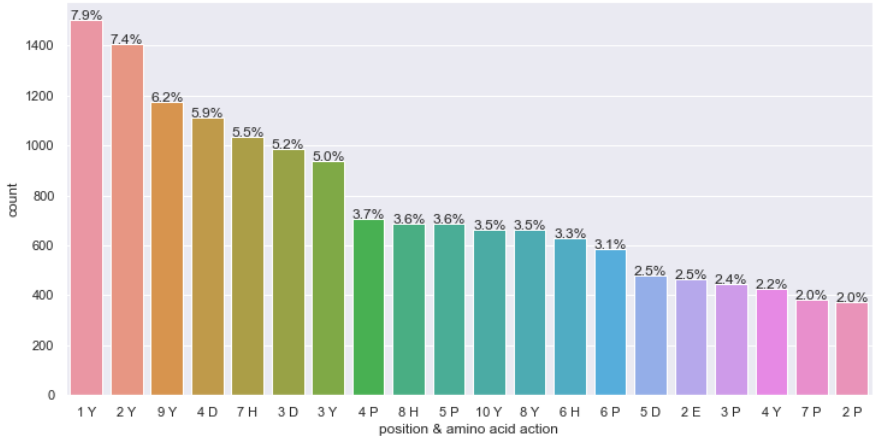




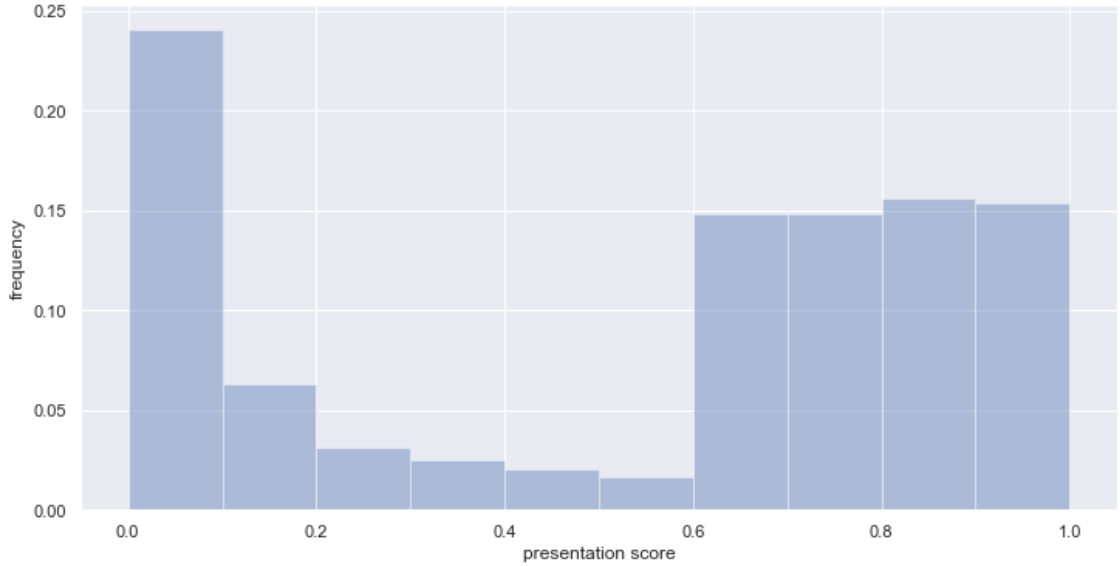
4. pos & amino action

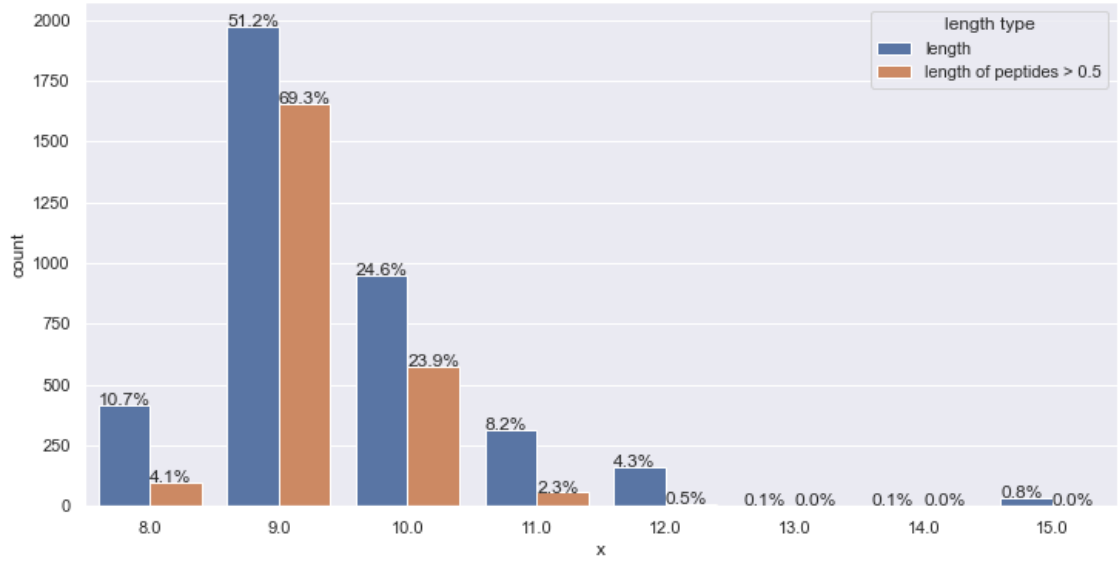






presentation score analysis: (53.6% optimized peptides have presentation scores > 0.5)





- Setting 1 (stop criteria = 0.6; sample rate = 80% from IEDB and 20% from random)

How to group human peptides into buckets using LSH?

human peptides: 197,850,667

Below are the first 3 peptides in the file:

> MAFSAEDVL
>
> AFSAEDVLK
>
> FSAEDVLKE
>
> .....

All the human peptides and virus peptides are extracted by sliding a 9-mer window along the human genome or Cov-19 virus.

Note that in our experiments, the distance we care about should be hamming distance instead of edit distance. Therefore, instead of simply counting the occurrences of *k*-mers, we need to take the position of *k*-mers into consideration.

However, 3 amino acid differences on hamming distance can significantly change the *k*-mers in the peptides. For example, "**X**AFS**X**EDV**X**" has 3 amino acid difference from the human peptide "MAFSAEDVL". However, these two peptides have only two common 3-mers among all the seven 3-mers, and 4 common 2-mers among all the 8 2-mers. With 3 amino acid differences, for the

human peptide "MAFSAEDVL", the above example "**X**AFS**X**EDV**X**" or "**X**AF**X**AEDV**X**" will change the common k-mers most significantly while the example "**XXX**SAEDVL" will lead to the least amount of change on the common k-mers.

For contiguous *3*-mers,

For discontiguous *k*-mers,

Inspired by the above analysis, the question is **how we can find a hashing function so that it can guarantee that peptides with less than 3 amino acid differences can be grouped into the same buckets?**

"**XXX**SAEDVL"

"**XX**F**X**AEDVL"

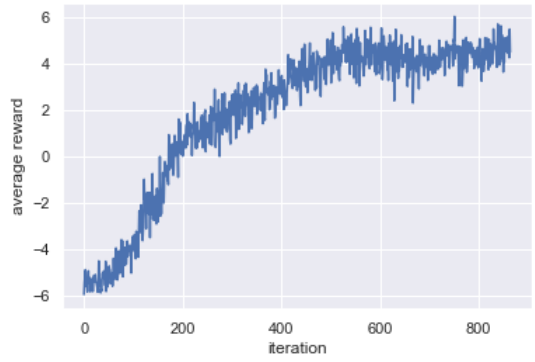
"**XX**FS**X**EDVL"

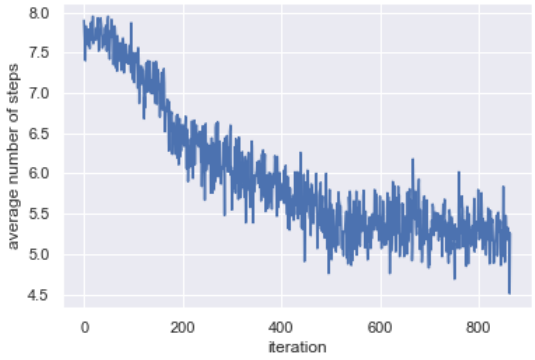
Update (5-26)

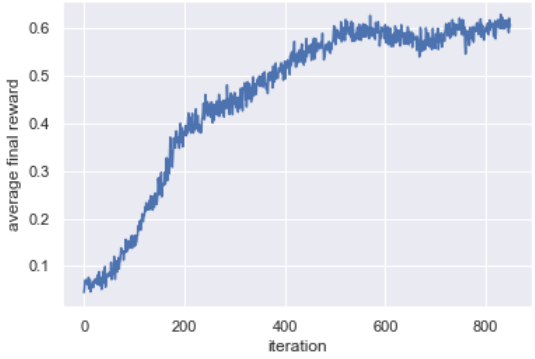
1. Result Analysis with longer running time and different stopping criteria

Experiment: stop criteria = 0.6

Setting 1 (sample rate = 50% from IEDB and 50% from random)

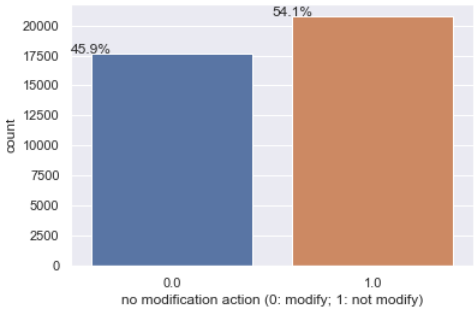


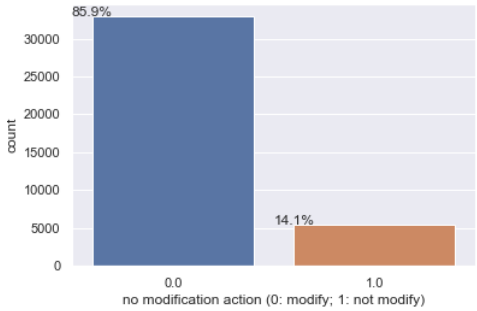


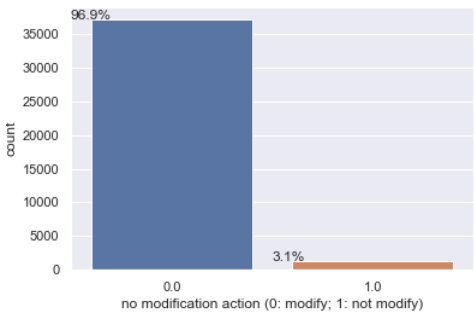


1. "No modification" action analysis

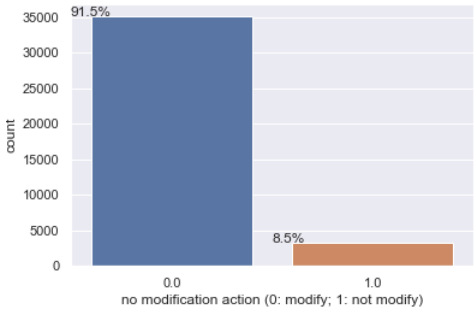
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

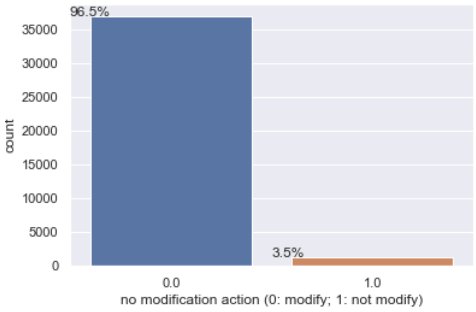




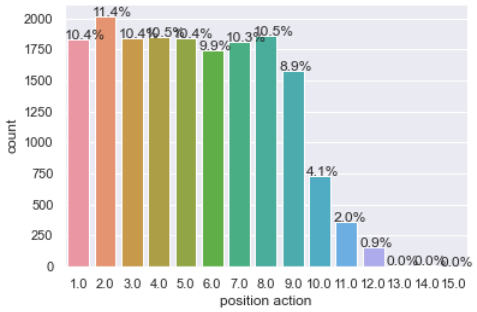


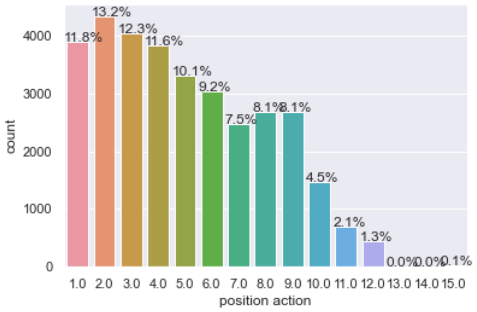
(10 iterations at 3/4) (last 10 iterations)

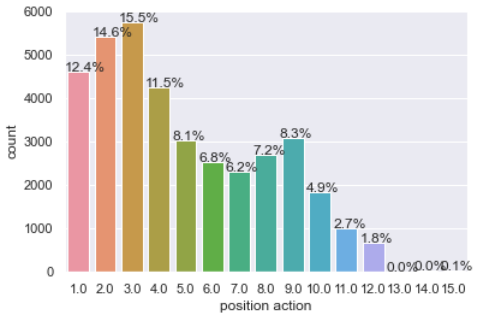


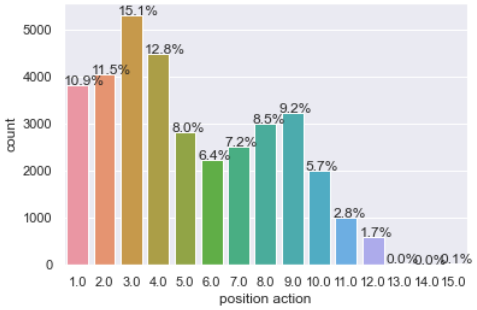


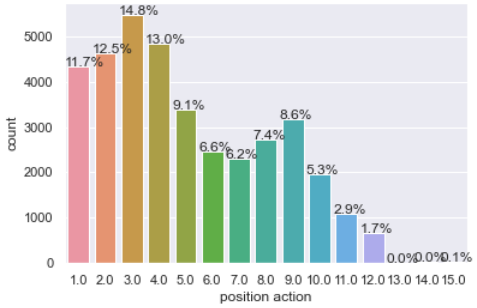
2. Position action analysis



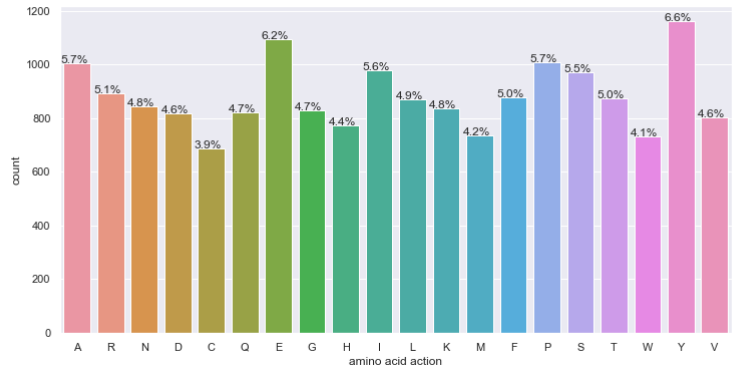


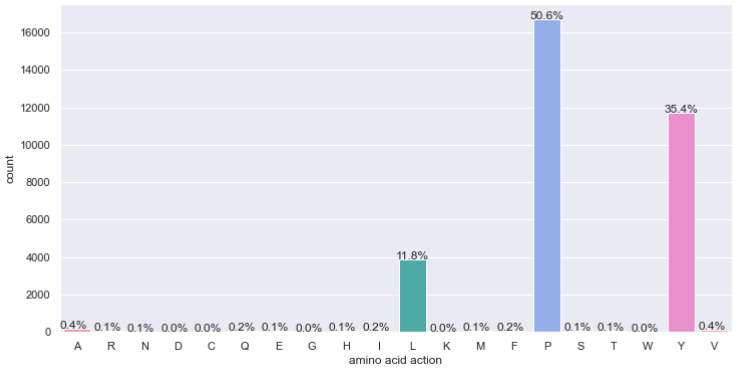


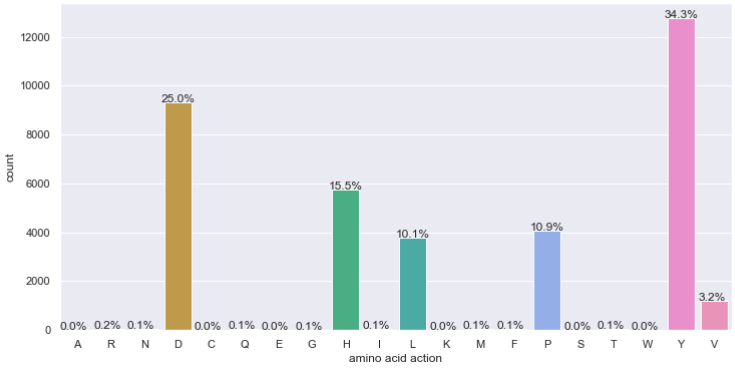


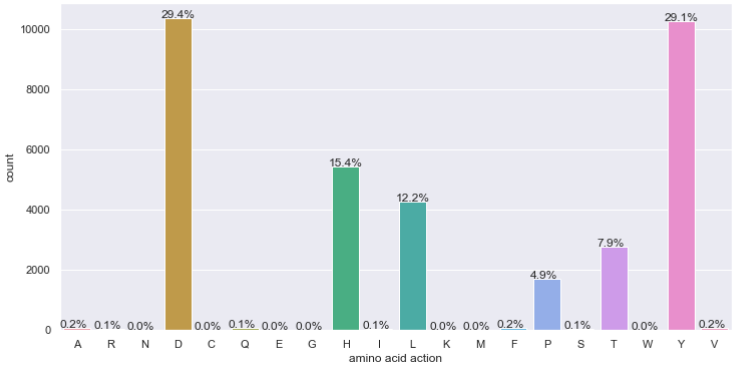


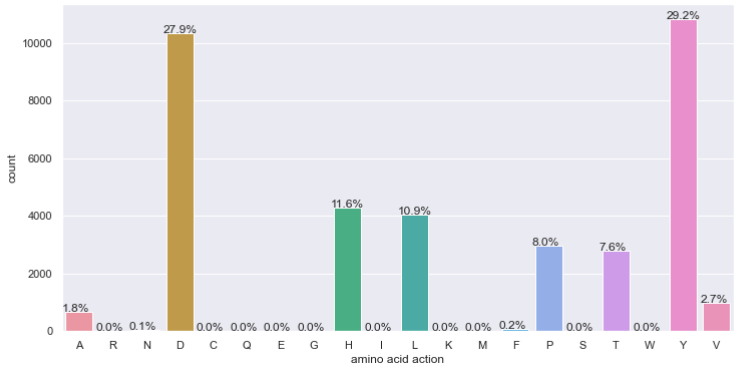
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.7% 5.1% 4.8% 4.6% 3.9% 4.7% 6.2% 4.7% 4.4% 5.6% 4.9% 4.8% 4.2% 5.0% 5.7% 5.5% 5.0% 4.1% 6.6% 4.6%

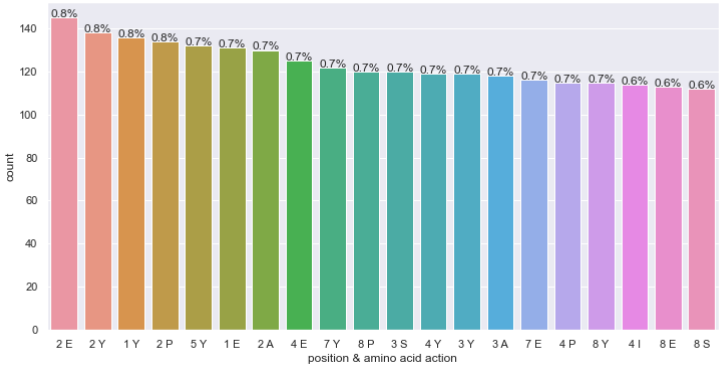

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.4% 0.1% 0.1% 0.0% 0.0% 0.2% 0.1% 0.0% 0.1% 0.2% 11.8% 0.0% 0.1% 0.2% 50.6% 0.1% 0.1% 0.0% 35.4% 0.4%

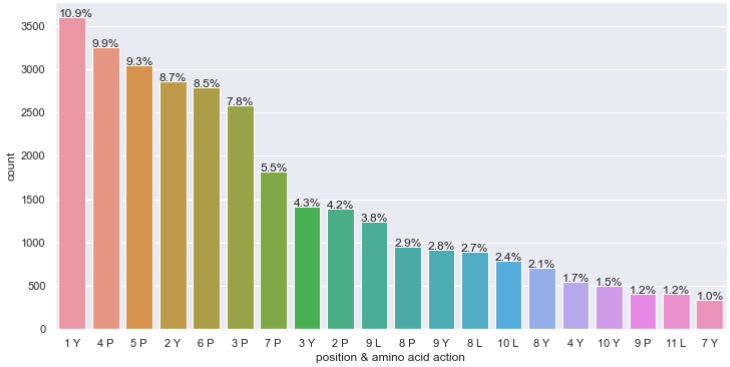

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.2% 0.1% 25.0% 0.0% 0.1% 0.0% 0.1% 15.5% 0.1% 10.1% 0.0% 0.1% 0.1% 10.9% 0.0% 0.1% 0.0% 34.3% 3.2%

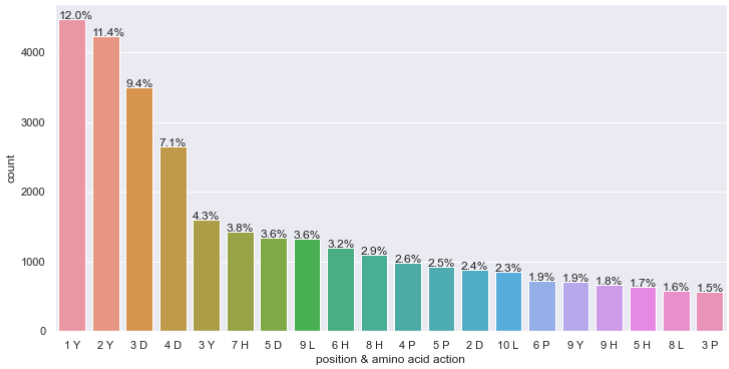

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.2% 0.1% 0.0% 29.4% 0.0% 0.1% 0.0% 0.0% 15.4% 0.1% 12.2% 0.0% 0.0% 0.2% 4.9% 0.1% 7.9% 0.0% 29.1% 0.2%

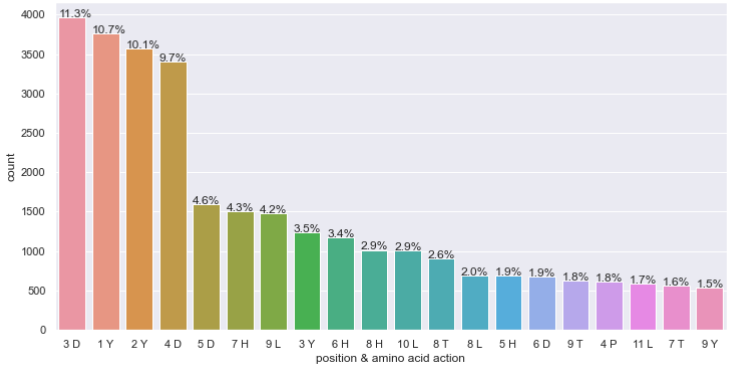


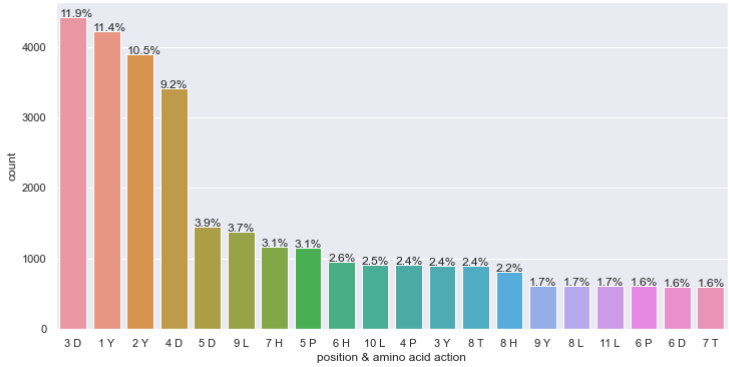
4. Position & Amino action analysis



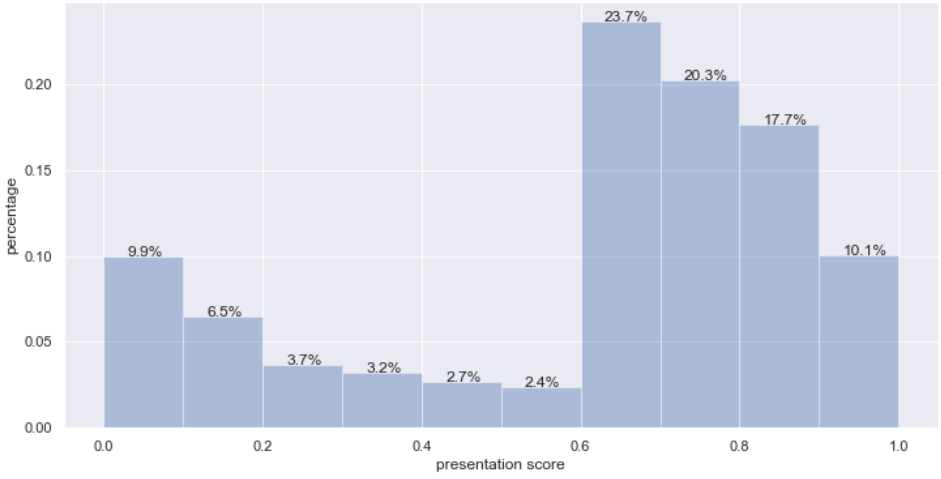




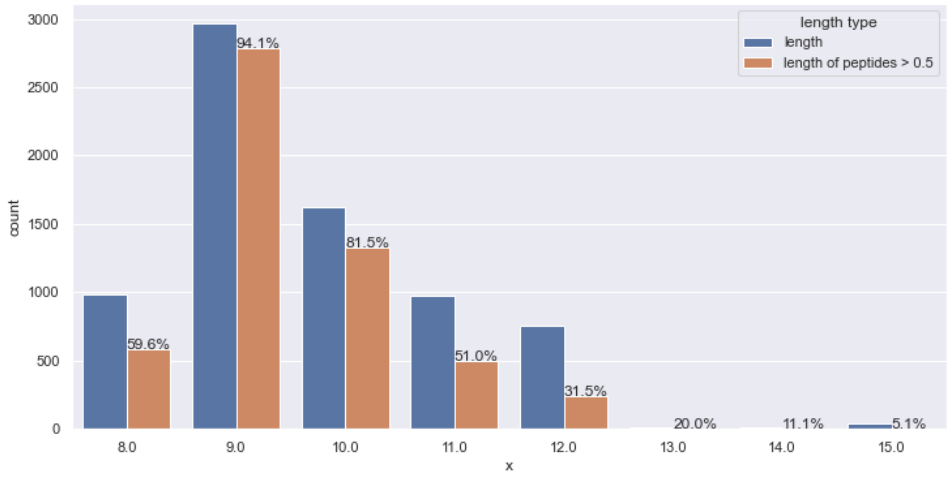




5. Reward analysis

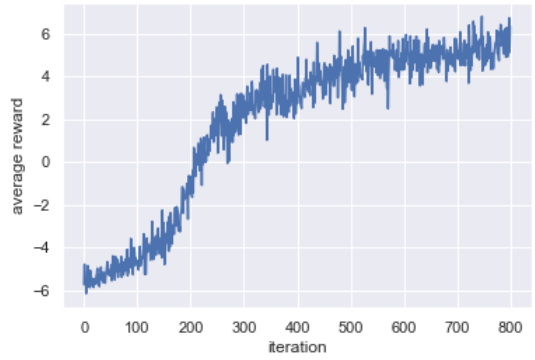


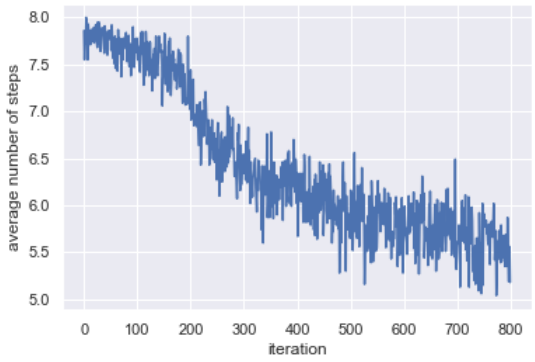
6. Length analysis

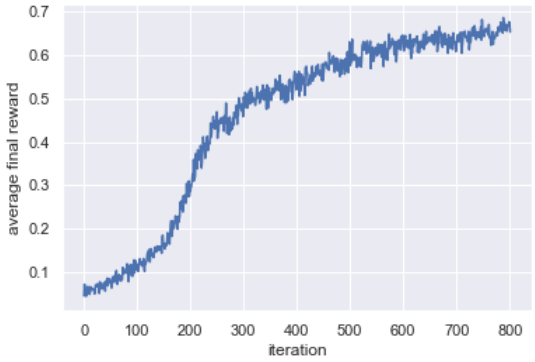


Experiment: stop criteria = 0.75

Setting 1 (sample rate = 50% from IEDB and 50% from random)

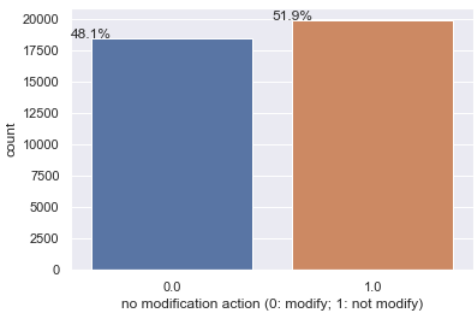


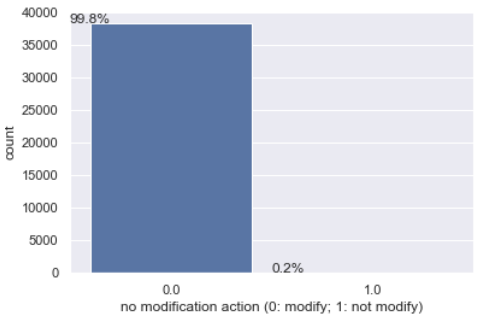


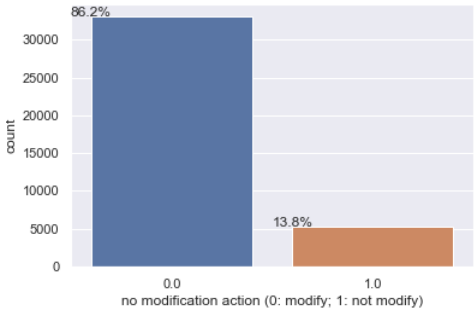


1. "No modification" action analysis

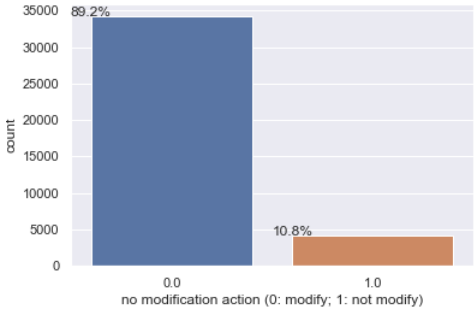
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

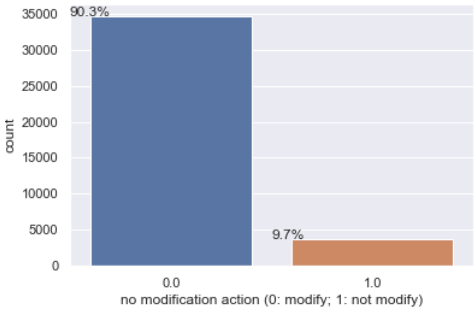




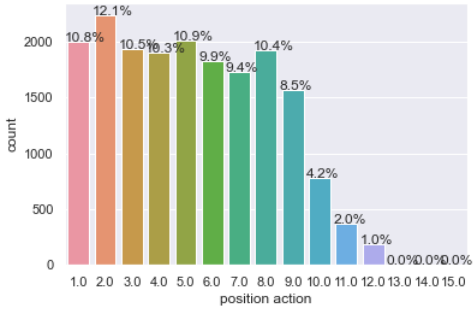


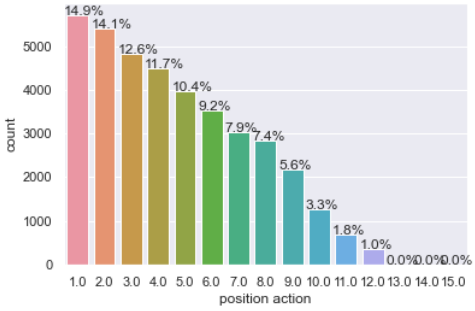
(10 iterations at 3/4) (last 10 iterations)

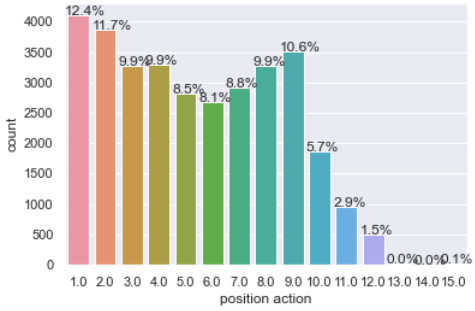


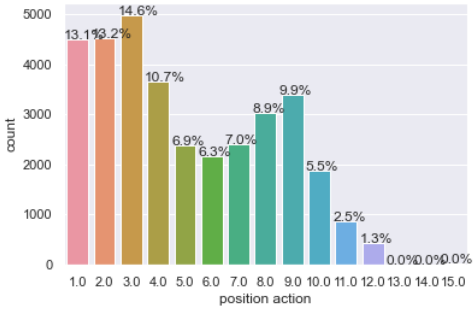


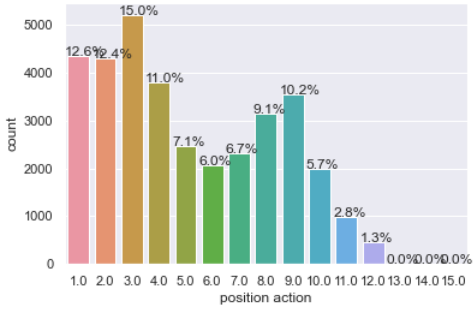
2. Position action analysis



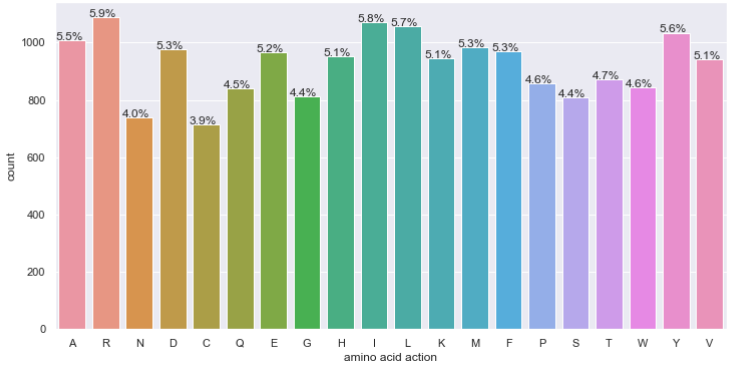


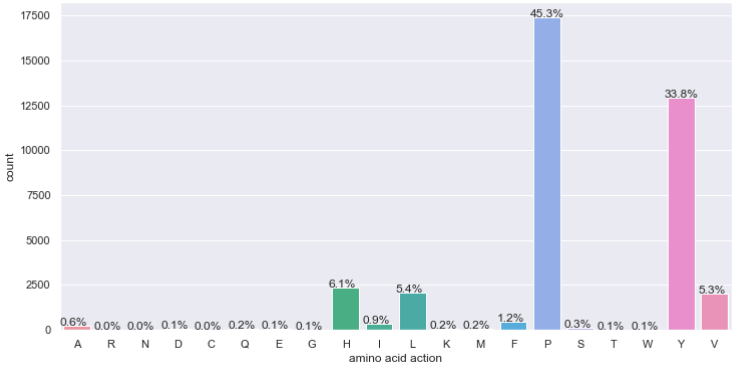


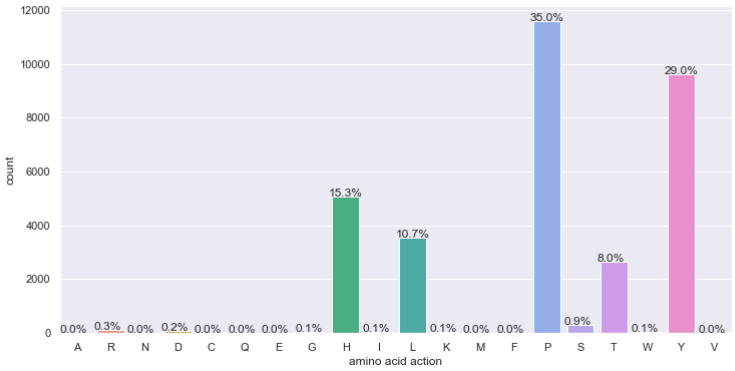


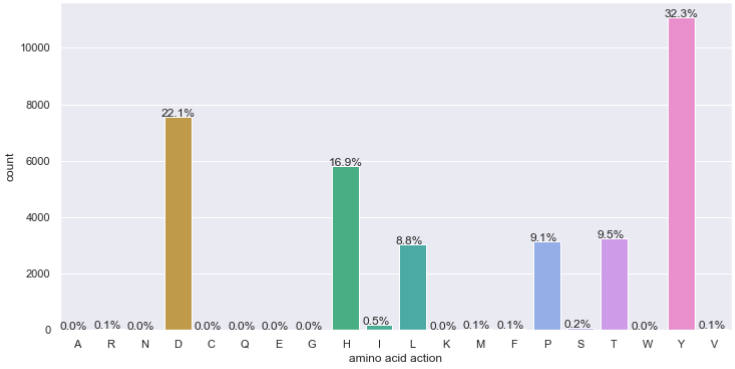


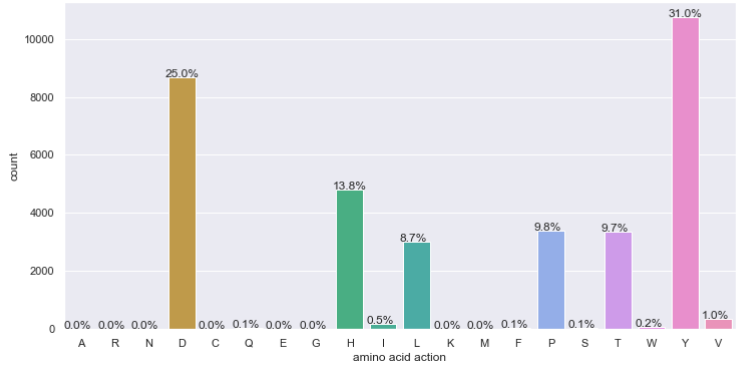
3. Amino action analysis



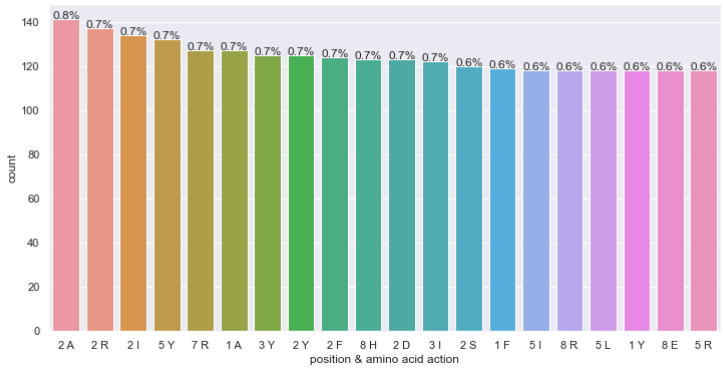


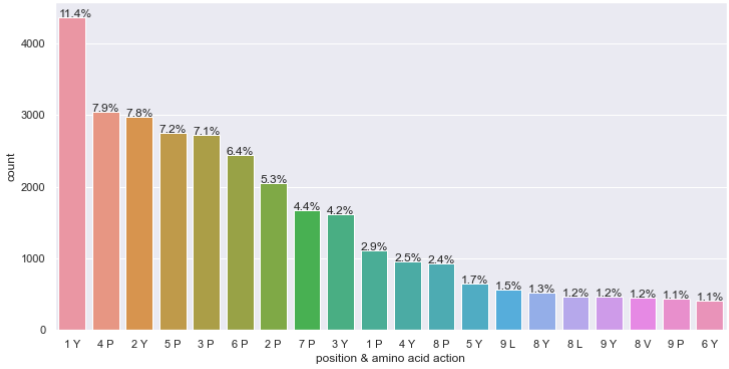


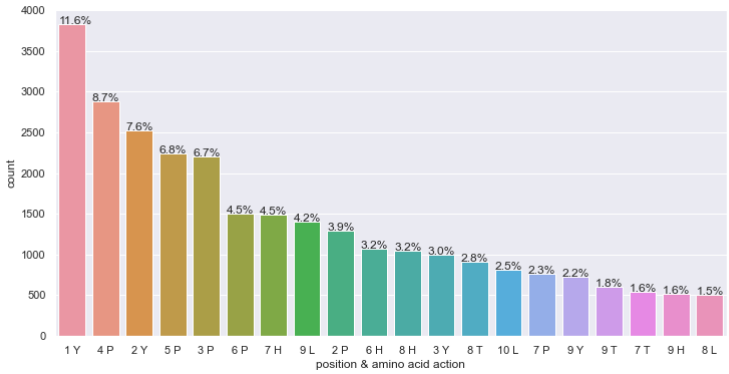


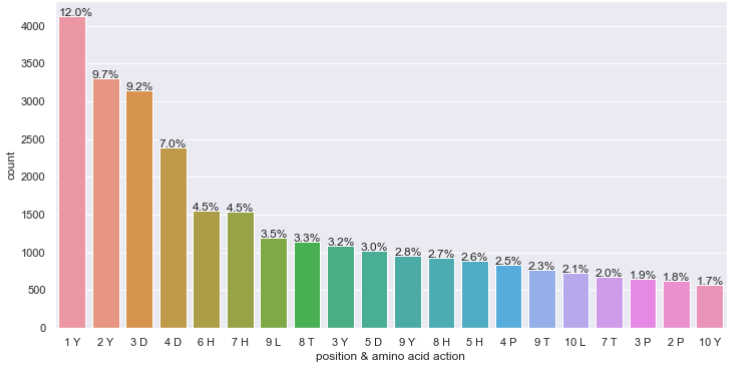


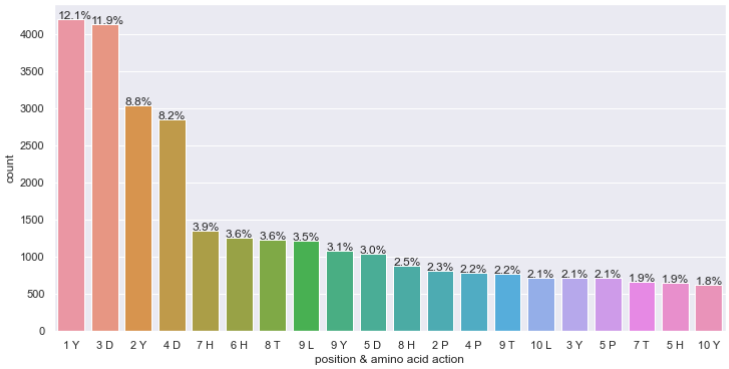
4. Position & Amino action analysis


count
position & amino acid action
0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6%
2 A 2 R 2 I 5 Y 7 R 1 A 3 Y 2 Y 2 F 8 H 2 D 3 I 2 S 1 F 5 I 8 R 5 L 1 Y 8 E 5 R

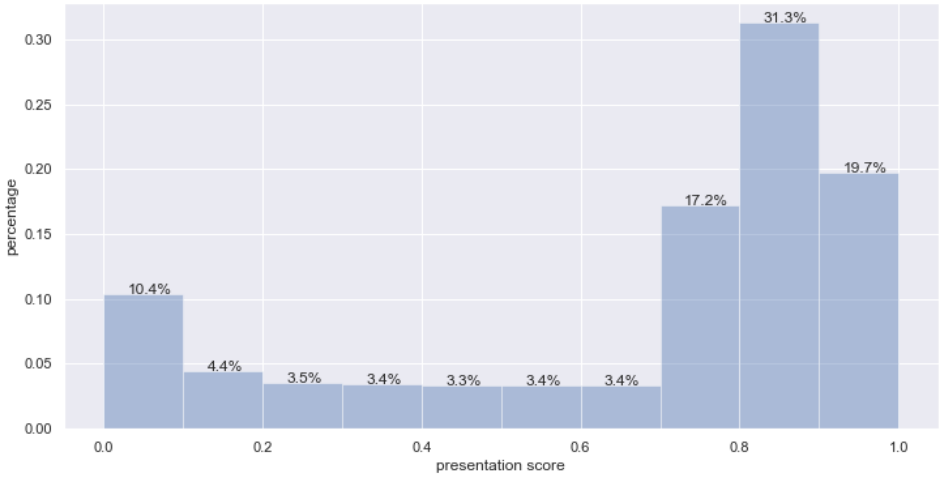

count
position & amino acid action
11.4% 7.9% 7.8% 7.2% 7.1% 6.4% 5.3% 4.4% 4.2% 2.9% 2.5% 2.4% 1.7% 1.5% 1.3% 1.2% 1.2% 1.2% 1.1% 1.1%
1 Y 4 P 2 Y 5 P 3 P 6 P 2 P 7 P 3 Y 1 P 4 Y 8 P 5 Y 9 L 8 Y 8 L 9 Y 8 V 9 P 6 Y


count
position & amino acid action
11.6% 8.7% 7.6% 6.8% 6.7% 4.5% 4.5% 4.2% 3.9% 3.2% 3.2% 3.0% 2.8% 2.5% 2.3% 2.2% 1.8% 1.6% 1.6% 1.5%
1 Y 4 P 2 Y 5 P 3 P 6 P 7 H 9 L 2 P 6 H 8 H 3 Y 8 T 10 L 7 P 9 Y 9 T 7 T 9 H 8 L

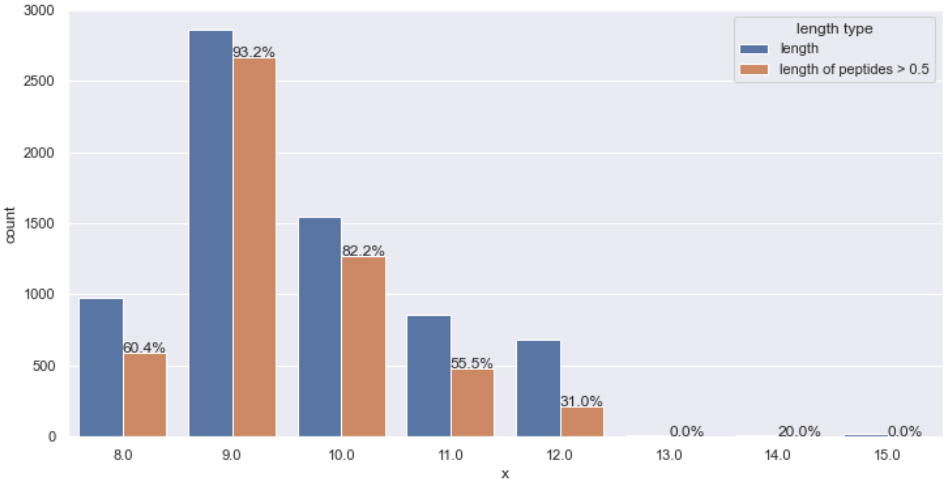

count
position & amino acid action
12.0% 9.7% 9.2% 7.0% 4.5% 4.5% 3.5% 3.3% 3.2% 3.0% 2.8% 2.7% 2.6% 2.5% 2.3% 2.1% 2.0% 1.9% 1.8% 1.7%
1 Y 2 Y 3 D 4 D 6 H 7 H 9 L 8 T 3 Y 5 D 9 Y 8 H 5 H 4 P 9 T 10 L 7 T 3 P 2 P 10 Y



5. Reward analysis

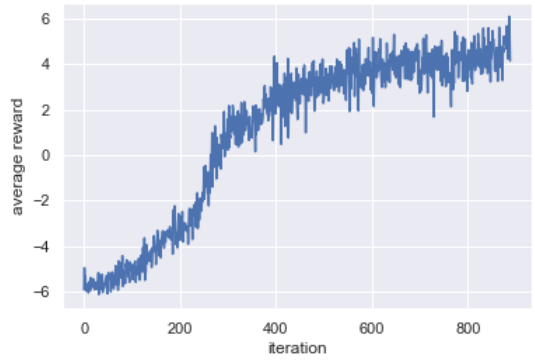


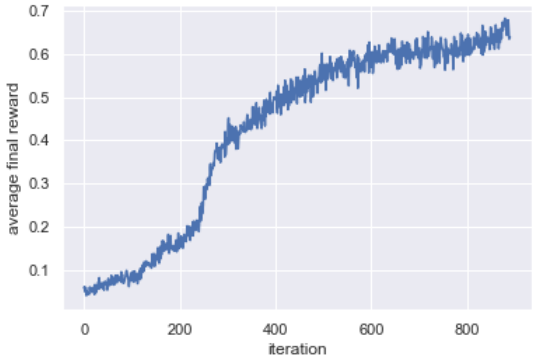
6. Length analysis



Experiment: stop criteria = 1.0

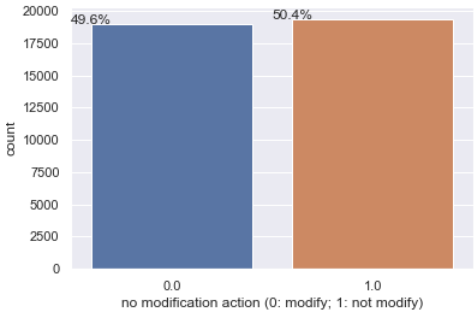
Setting 1 (sample rate = 50% from IEDB and 50% from random)

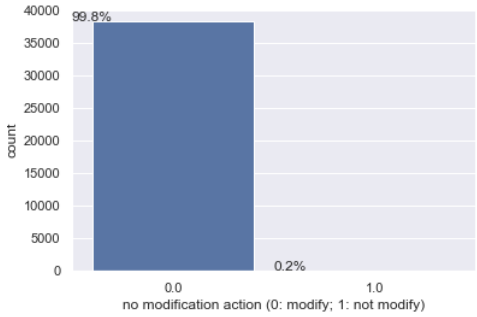


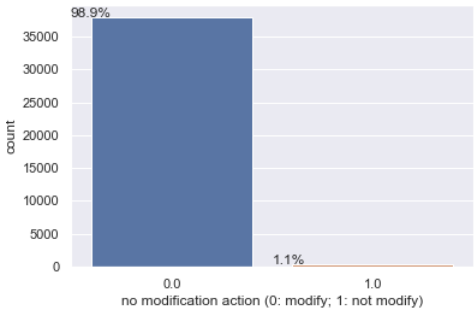


1. "No modification" action analysis

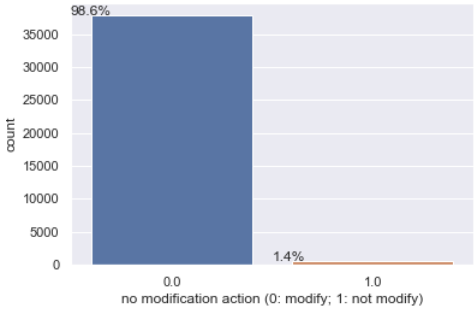
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

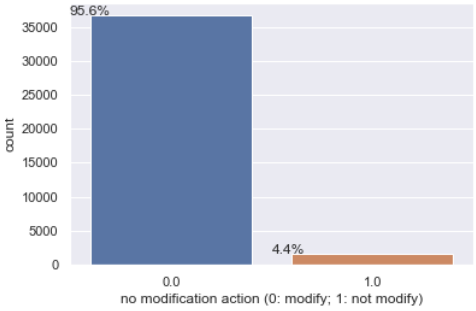




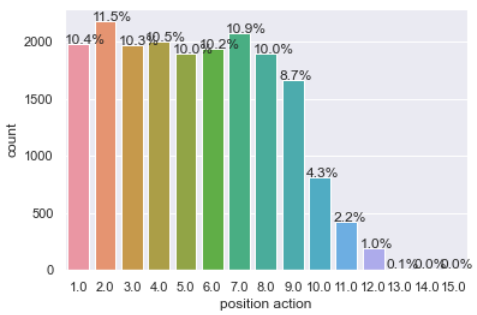


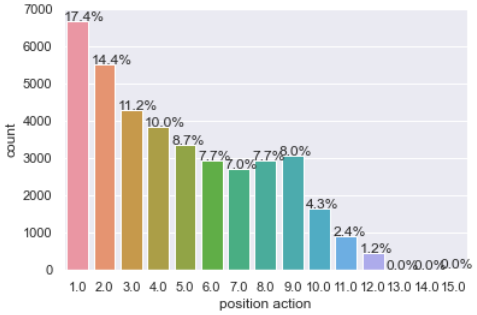
(10 iterations at 3/4) (last 10 iterations)

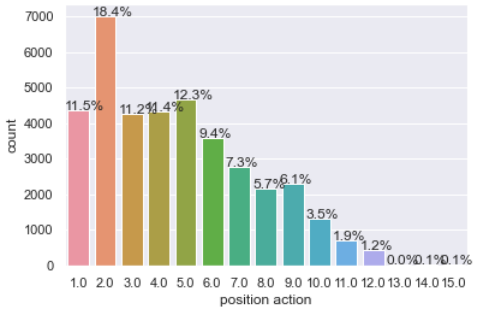


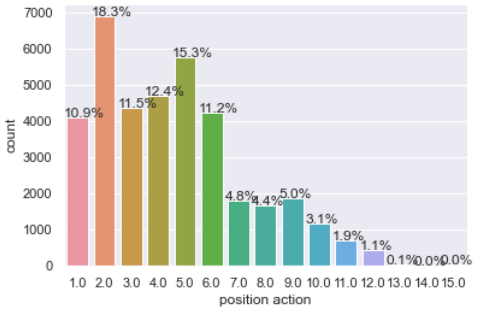


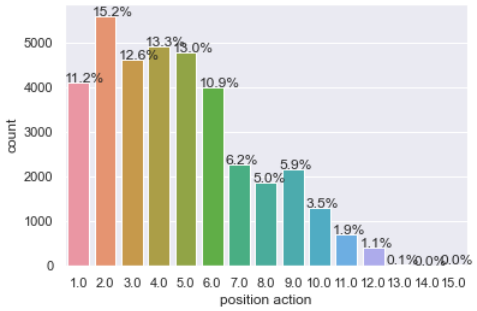
2. Position action analysis



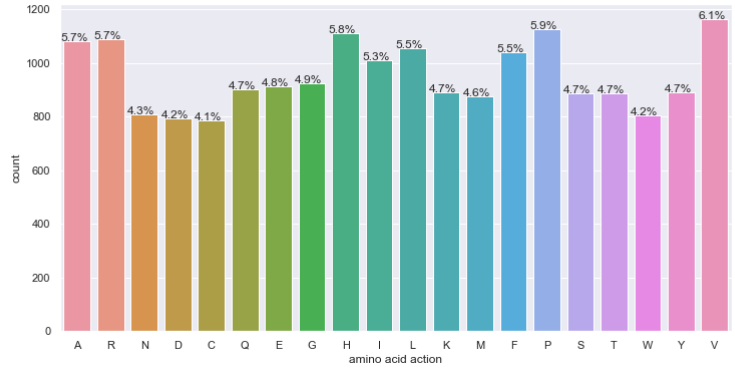


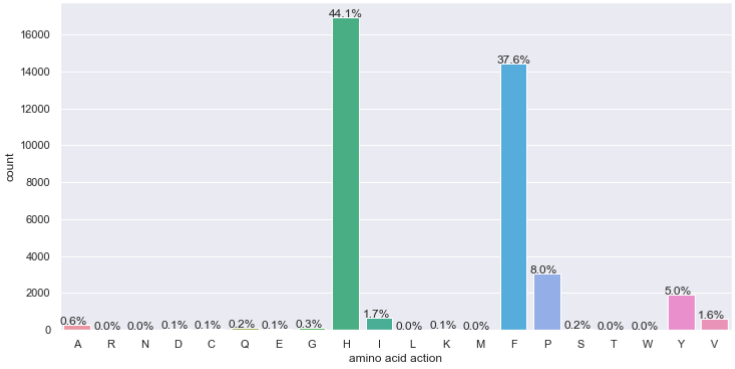


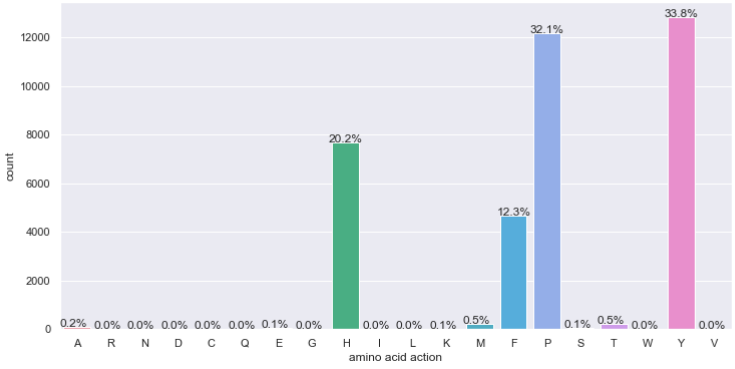


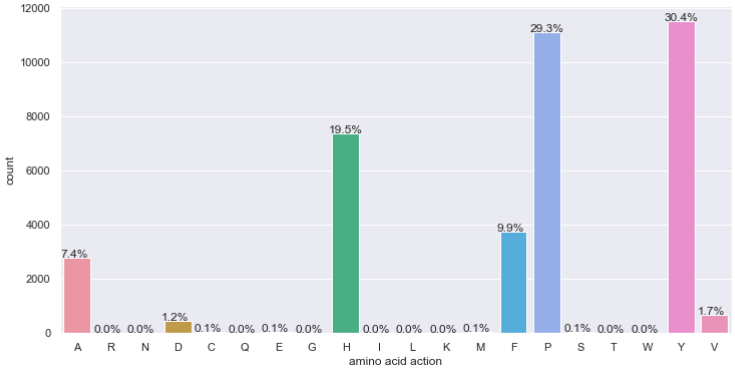


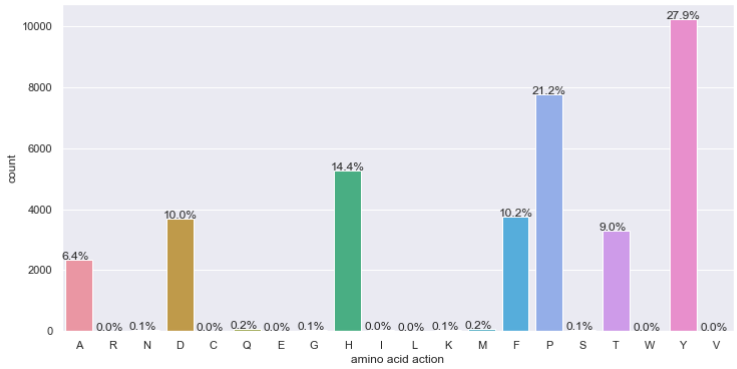
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.7% 5.7% 4.3% 4.2% 4.1% 4.7% 4.8% 4.9% 5.8% 5.3% 5.5% 4.7% 4.6% 5.5% 5.9% 4.7% 4.7% 4.2% 4.7% 6.1%

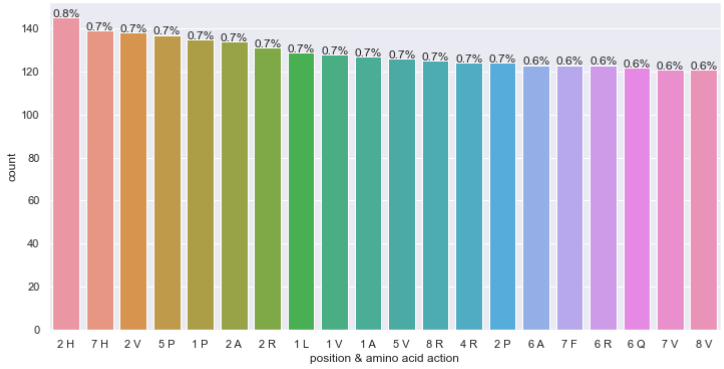

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.6% 0.0% 0.0% 0.1% 0.1% 0.2% 0.1% 0.3% 44.1% 1.7% 0.0% 0.1% 0.0% 37.6% 8.0% 0.2% 0.0% 0.0% 5.0% 1.6%

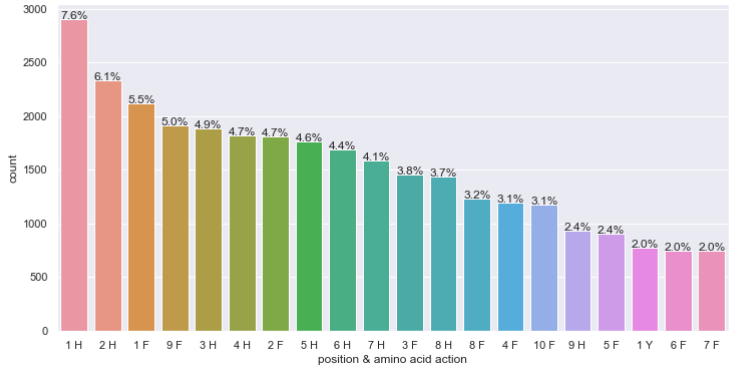

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.2% 0.0% 0.0% 0.0% 0.0% 0.0% 0.1% 0.0% 20.2% 0.0% 0.0% 0.1% 0.5% 12.3% 32.1% 0.1% 0.5% 0.0% 33.8% 0.0%

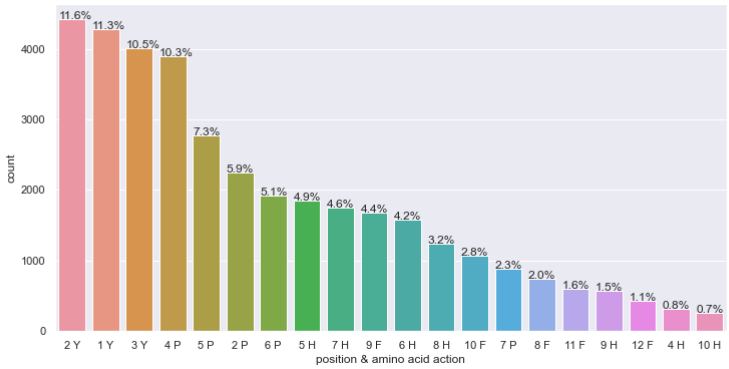

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
7.4% 0.0% 0.0% 1.2% 0.1% 0.0% 0.1% 0.0% 19.5% 0.0% 0.0% 0.0% 0.1% 9.9% 29.3% 0.1% 0.0% 0.0% 30.4% 1.7%

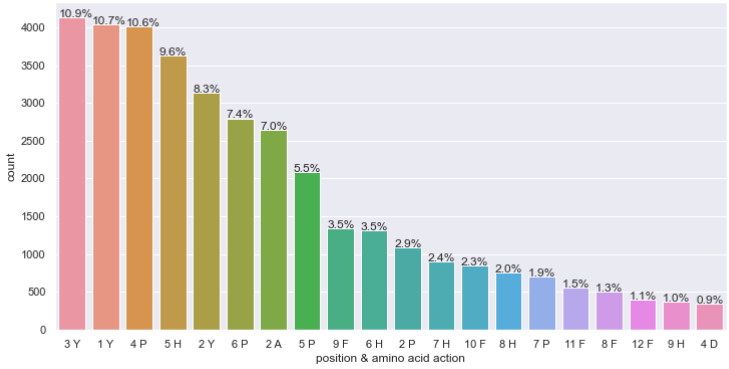


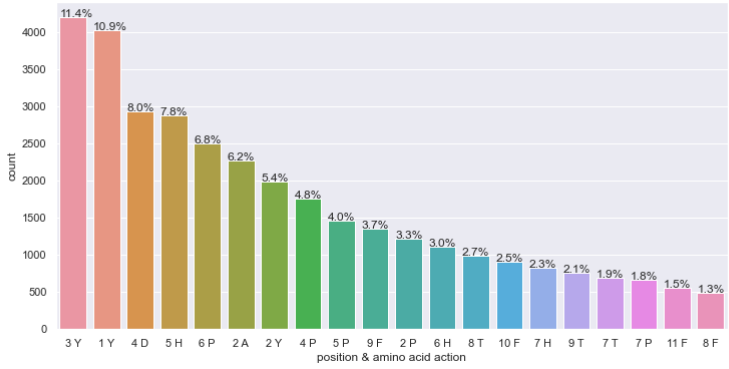
4. Position & Amino action analysis



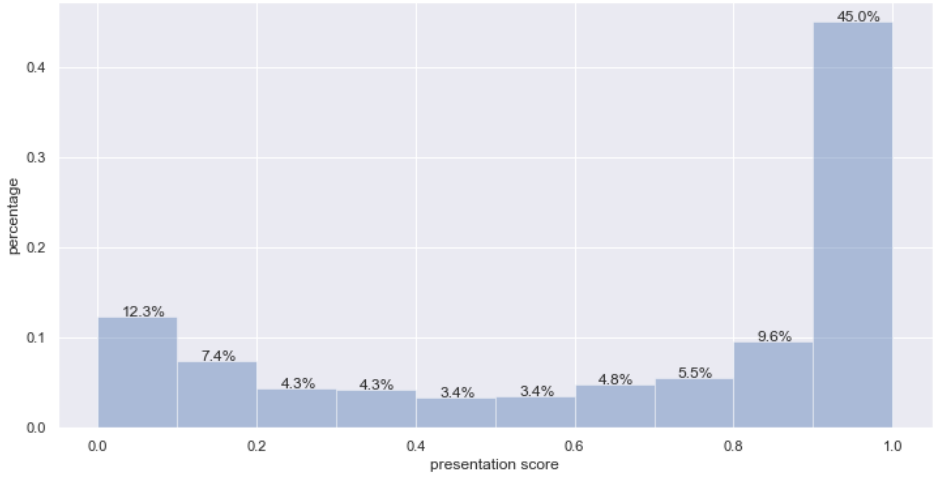




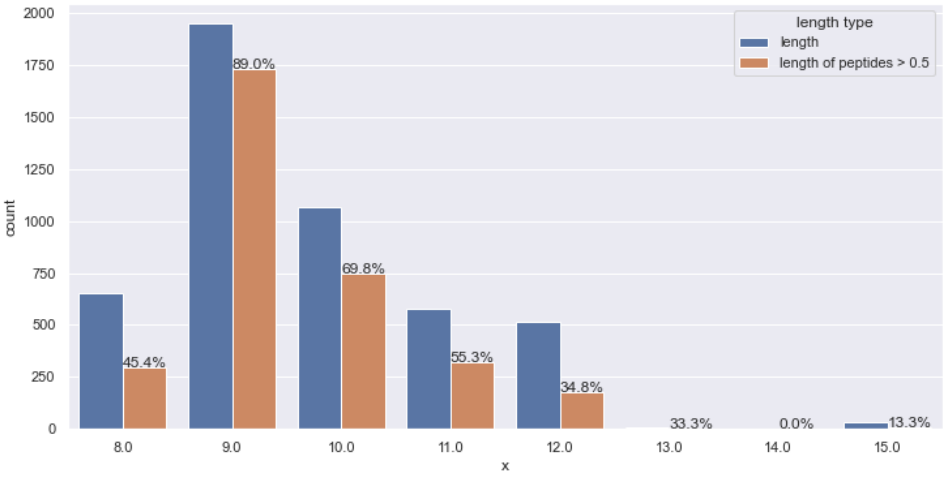




5. Reward analysis



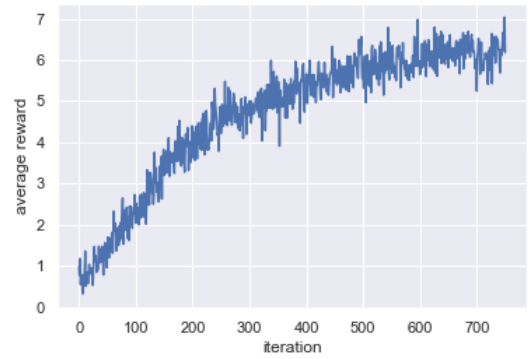
6. Length analysis

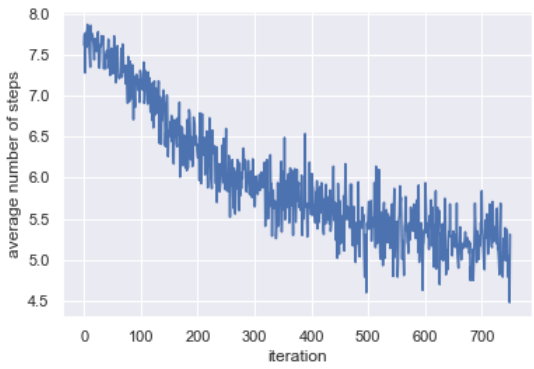


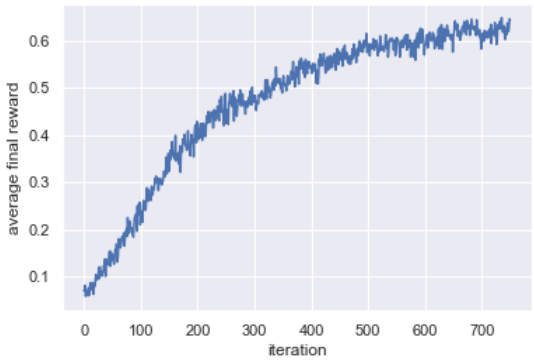
2. Result Analysis with only final rewards

Experiment: stop criteria = 0.6

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

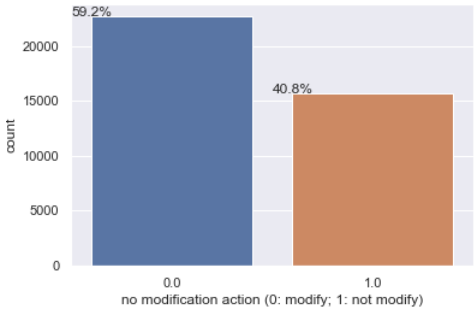
- 

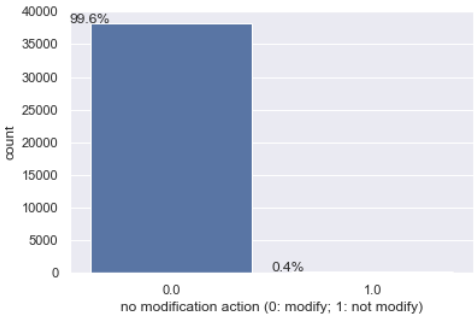


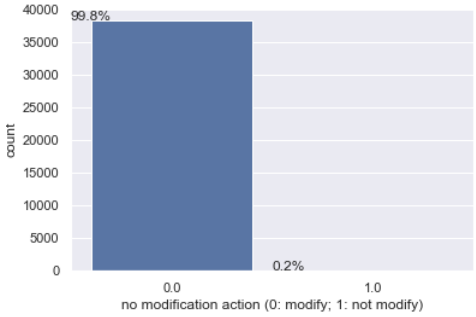


1. "No modification" action analysis

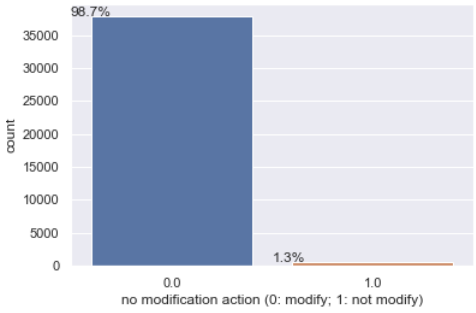
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

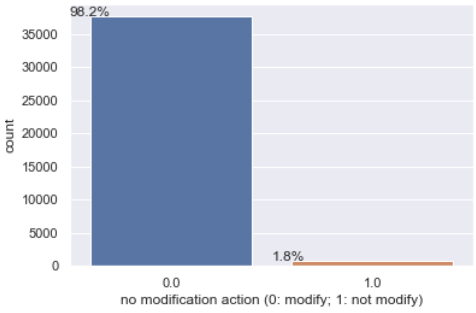




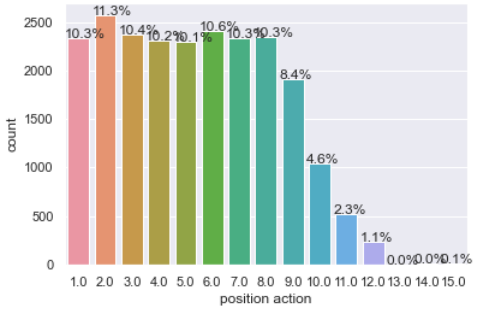


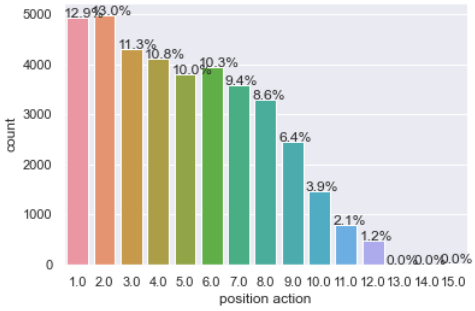
(10 iterations at 3/4) (last 10 iterations)

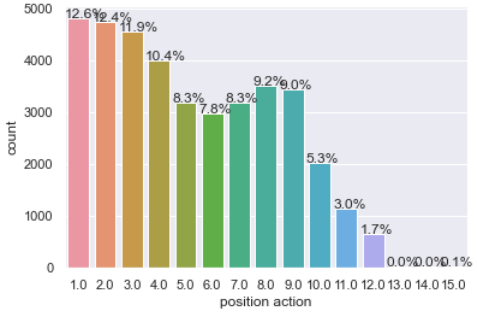


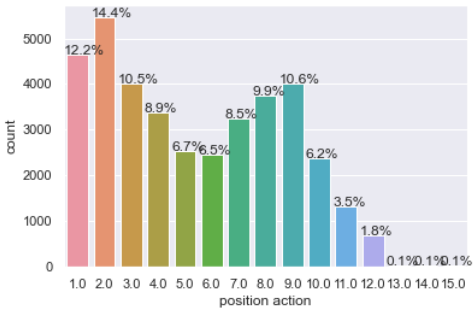


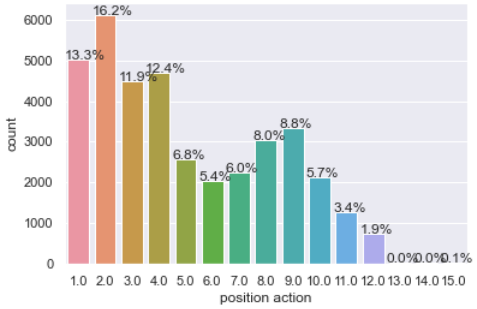
2. Position action analysis



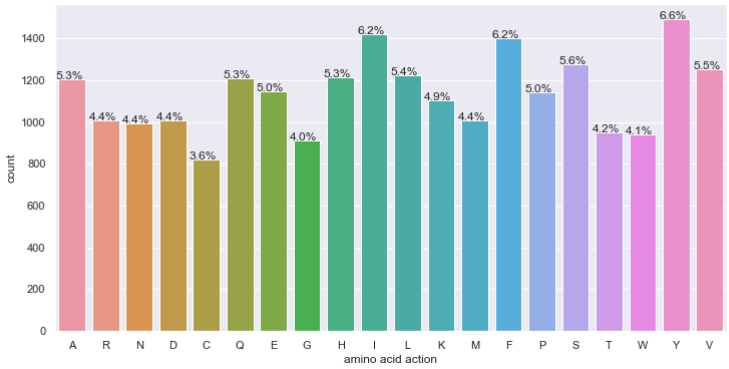


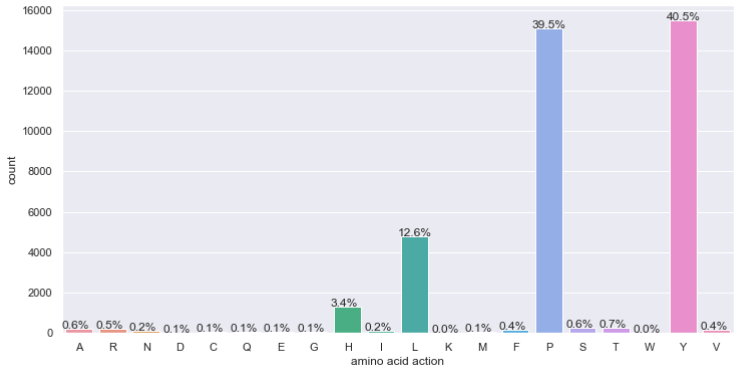


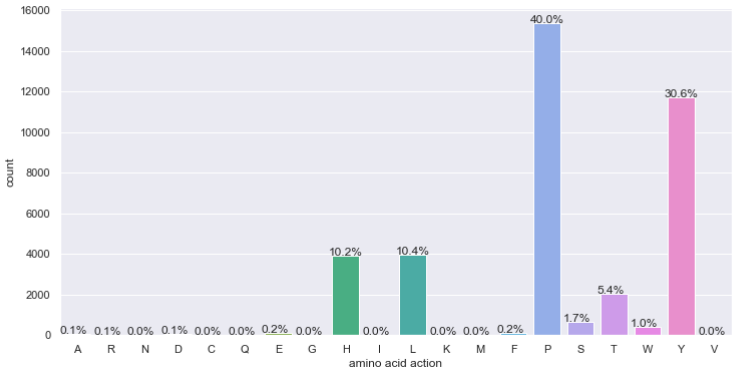


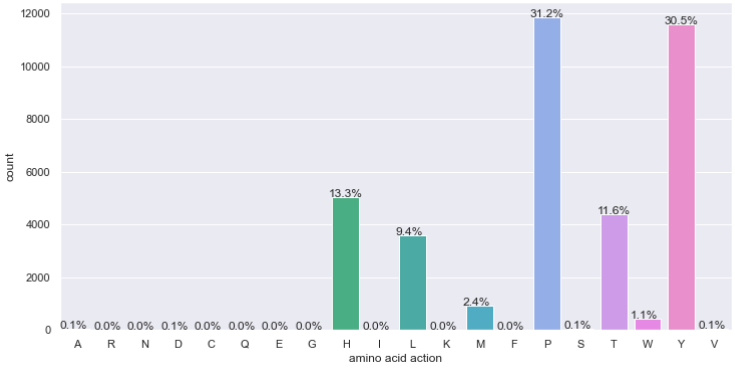


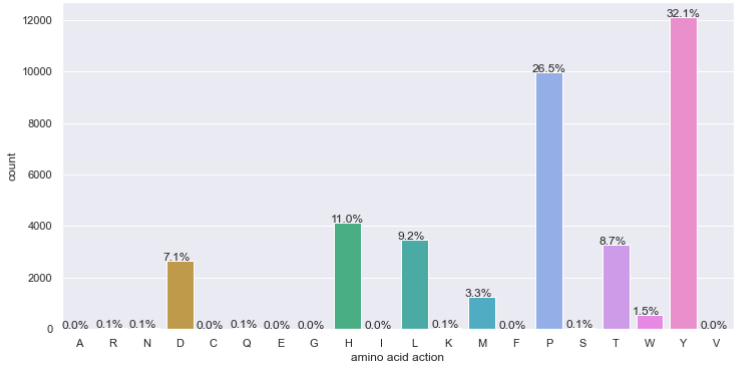
3. Amino action analysis



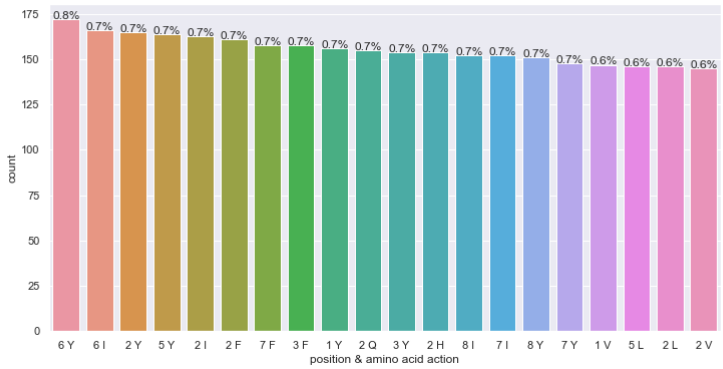


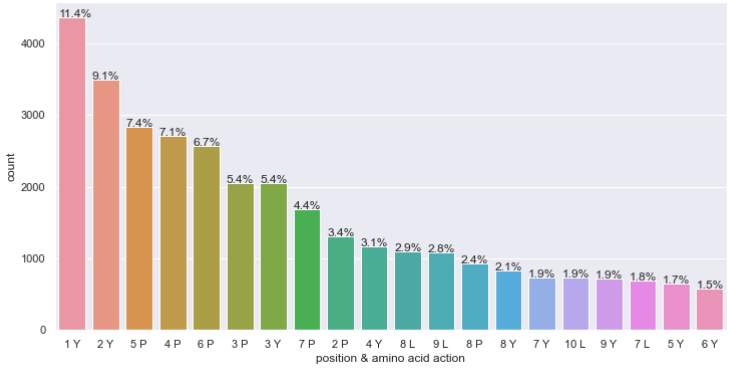


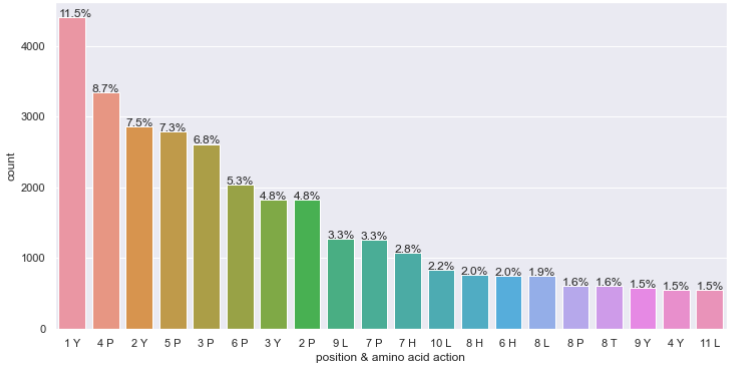


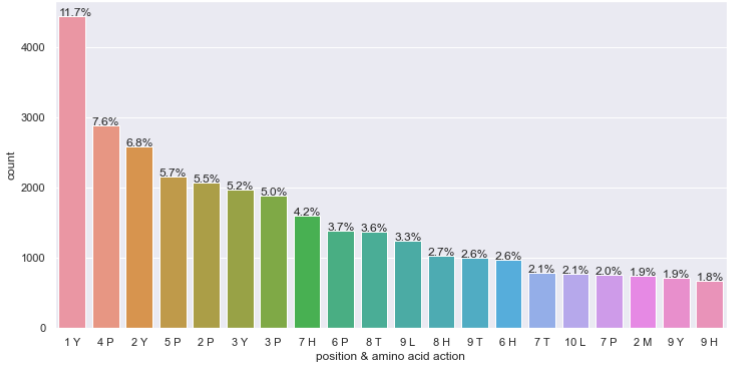


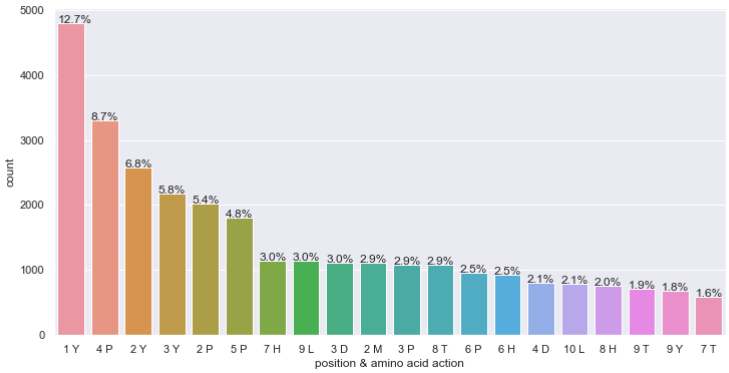
4. Position & Amino action analysis


count
position & amino acid action
0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.6% 0.6% 0.6% 0.6%
6 Y 6 I 2 Y 5 Y 2 I 2 F 7 F 3 F 1 Y 2 Q 3 Y 2 H 8 I 7 I 8 Y 7 Y 1 V 5 L 2 L 2 V

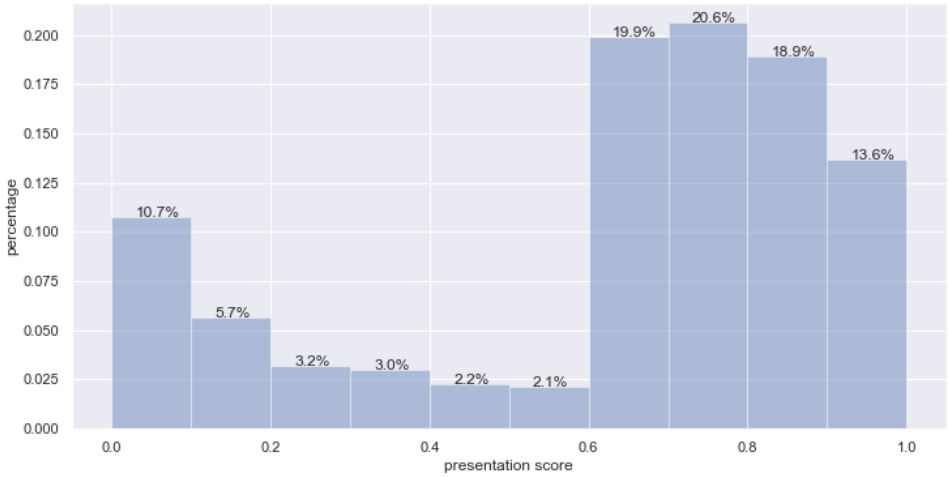

count
position & amino acid action
11.4% 9.1% 7.4% 7.1% 6.7% 5.4% 5.4% 4.4% 3.4% 3.1% 2.9% 2.8% 2.4% 2.1% 1.9% 1.9% 1.9% 1.8% 1.7% 1.5%
1 Y 2 Y 5 P 4 P 6 P 3 P 3 Y 7 P 2 P 4 Y 8 L 9 L 8 P 8 Y 7 Y 10 L 9 Y 7 L 5 Y 6 Y


count
position & amino acid action
11.5% 8.7% 7.5% 7.3% 6.8% 5.3% 4.8% 4.8% 3.3% 3.3% 2.8% 2.2% 2.0% 2.0% 1.9% 1.6% 1.6% 1.5% 1.5% 1.5%
1 Y 4 P 2 Y 5 P 3 P 6 P 3 Y 2 P 9 L 7 P 7 H 10 L 8 H 6 H 8 L 8 P 8 T 9 Y 4 Y 11 L

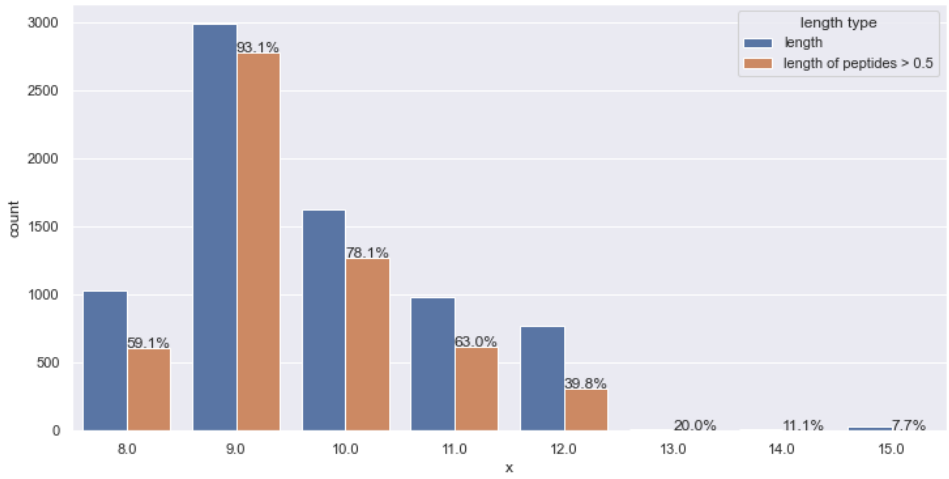

count
position & amino acid action
11.7% 7.6% 6.8% 5.7% 5.5% 5.2% 5.0% 4.2% 3.7% 3.6% 3.3% 2.7% 2.6% 2.6% 2.1% 2.1% 2.0% 1.9% 1.9% 1.8%
1 Y 4 P 2 Y 5 P 2 P 3 Y 3 P 7 H 6 P 8 T 9 L 8 H 9 T 6 H 7 T 10 L 7 P 2 M 9 Y 9 H



5. Reward analysis

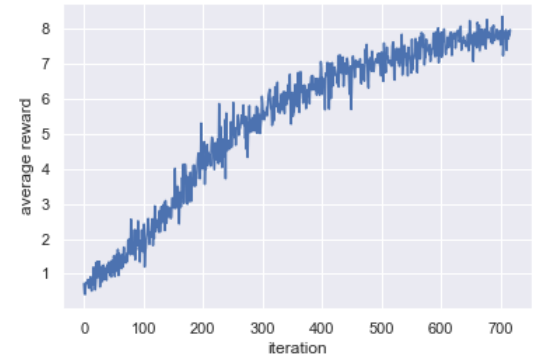


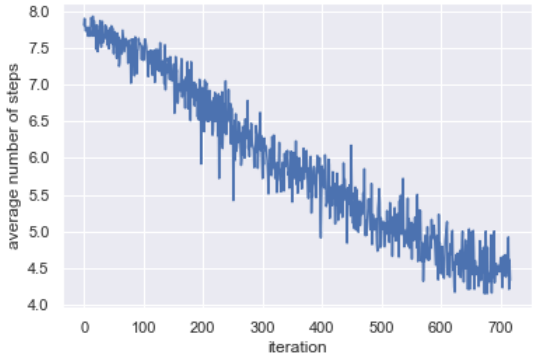
6. Length analysis

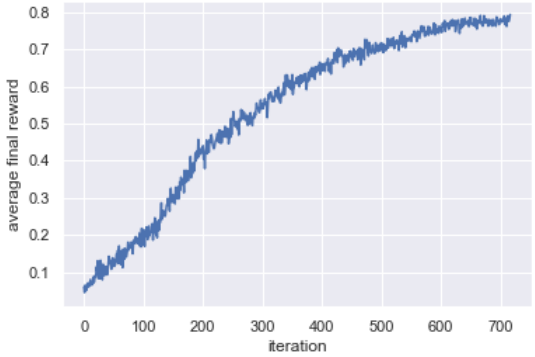


Experiment: stop criteria = 0.75

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

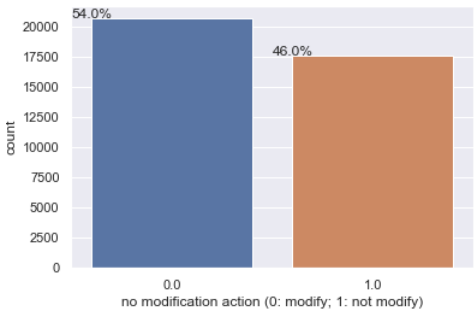


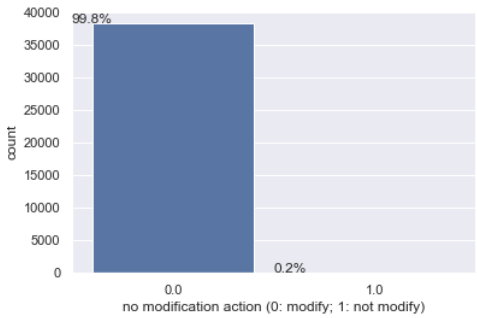


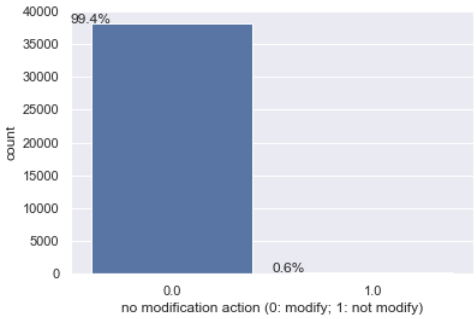


1. "No modification" action analysis

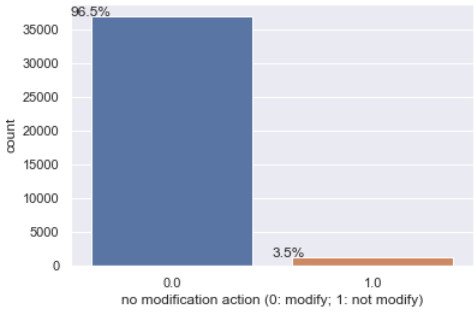
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

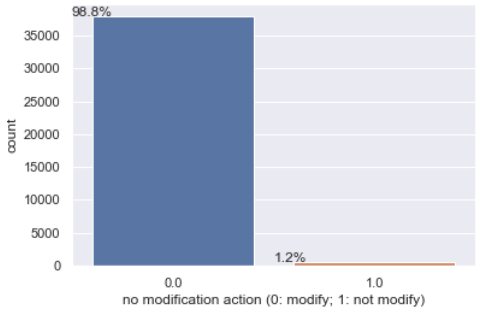




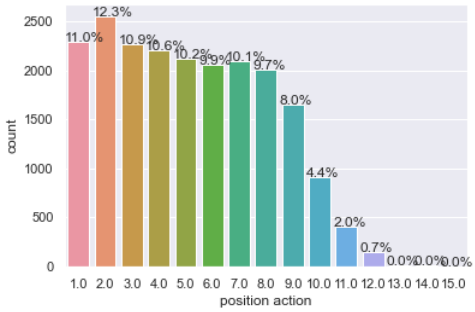


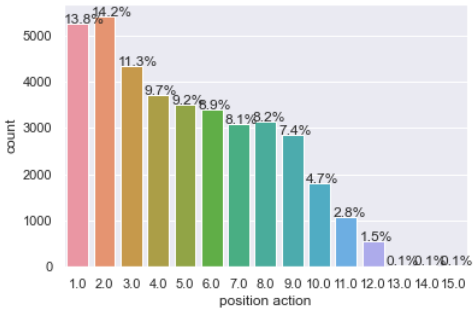
(10 iterations at 3/4) (last 10 iterations)

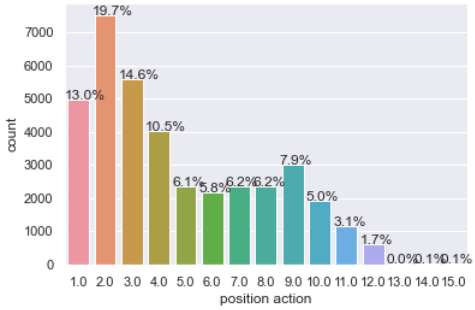


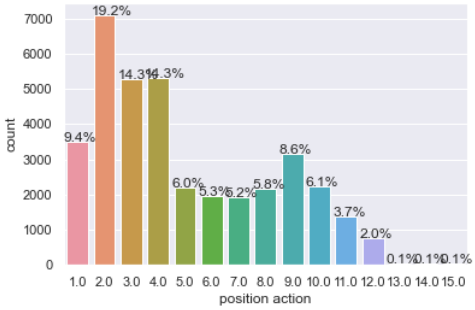


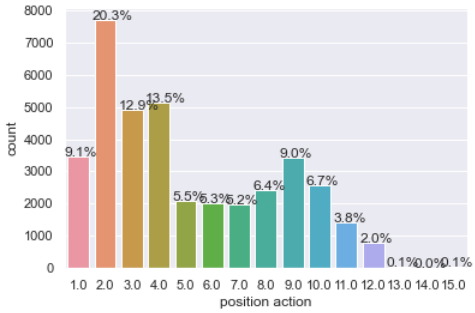
2. Position action analysis



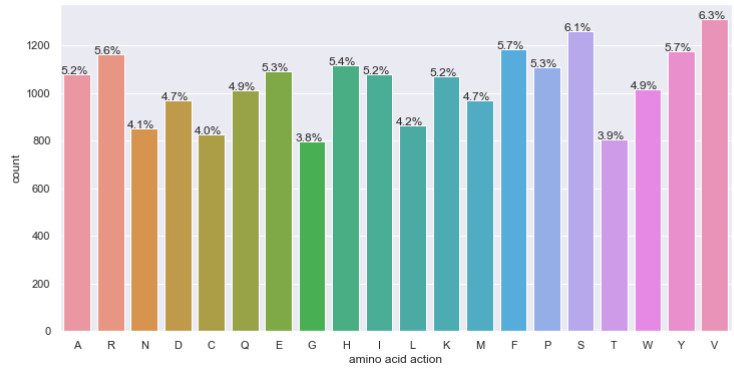


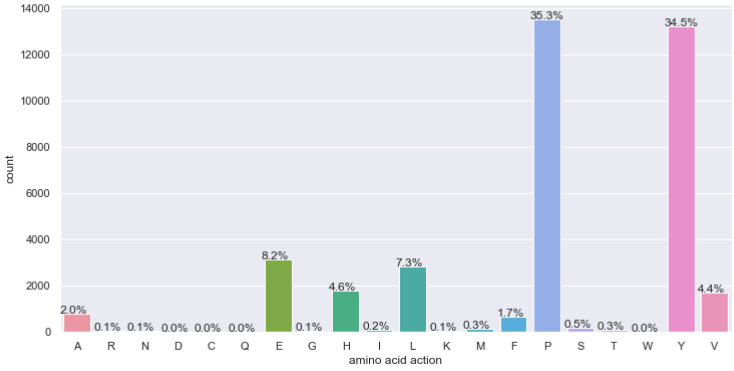


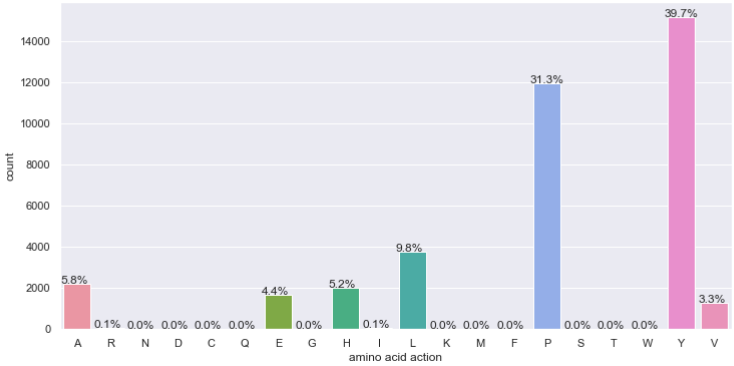


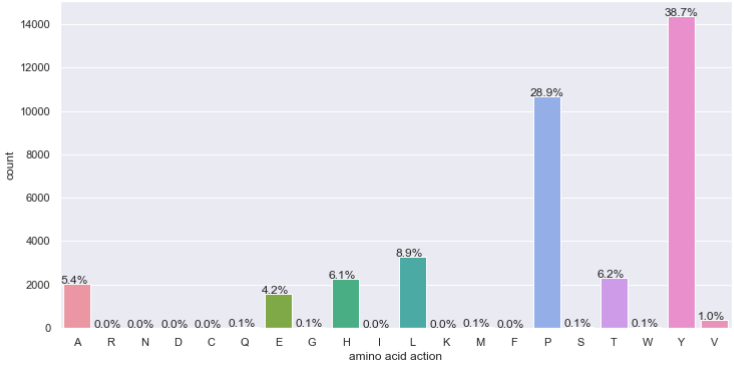


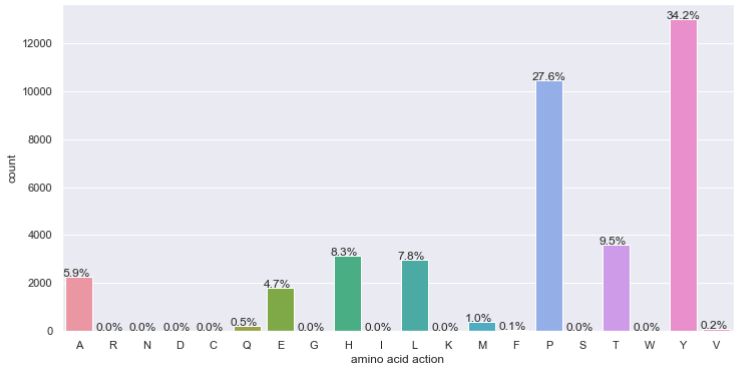
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.2% 5.6% 4.1% 4.7% 4.0% 4.9% 5.3% 3.8% 5.4% 5.2% 4.2% 5.2% 4.7% 5.7% 5.3% 6.1% 3.9% 4.9% 5.7% 6.3%

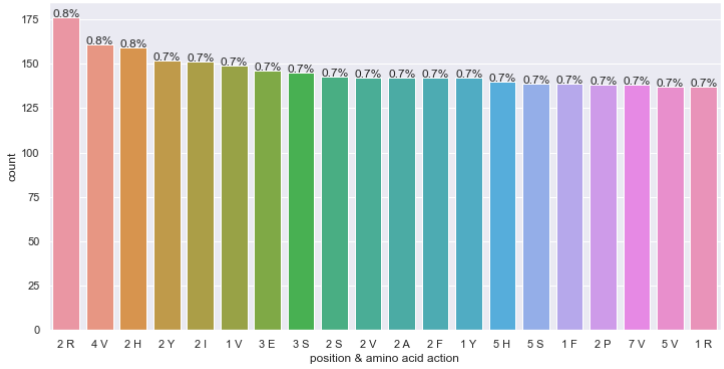

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
2.0% 0.1% 0.1% 0.0% 0.0% 0.0% 8.2% 0.1% 4.6% 0.2% 7.3% 0.1% 0.3% 1.7% 35.3% 0.5% 0.3% 0.0% 34.5% 4.4%

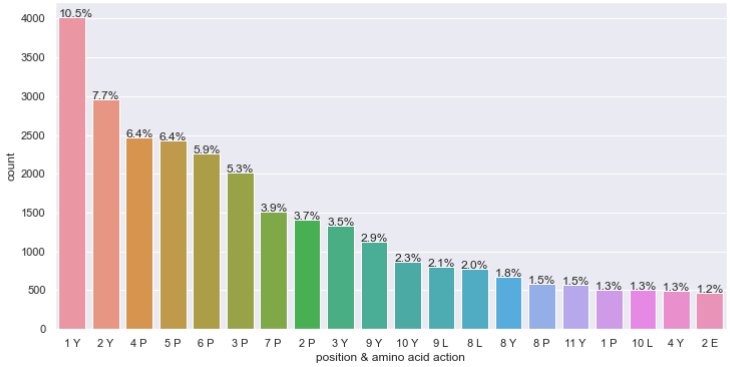

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.8% 0.1% 0.0% 0.0% 0.0% 0.0% 4.4% 0.0% 5.2% 0.1% 9.8% 0.0% 0.0% 0.0% 31.3% 0.0% 0.0% 0.0% 39.7% 3.3%

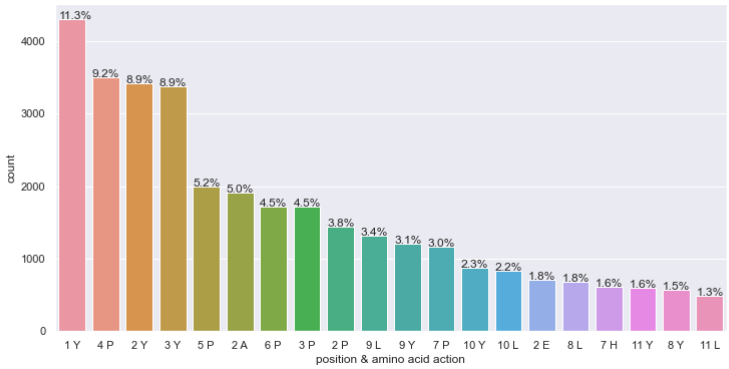

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.4% 0.0% 0.0% 0.0% 0.0% 0.1% 4.2% 0.1% 6.1% 0.0% 8.9% 0.0% 0.1% 0.0% 28.9% 0.1% 6.2% 0.1% 38.7% 1.0%

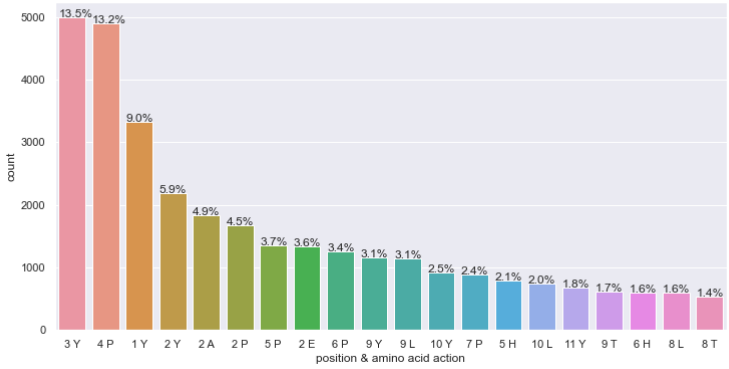


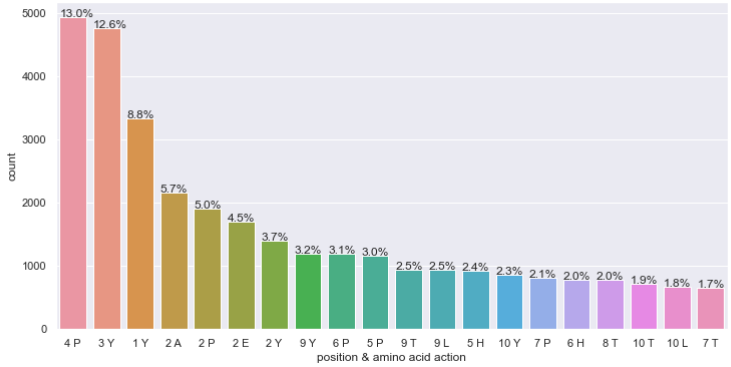
4. Position & Amino action analysis



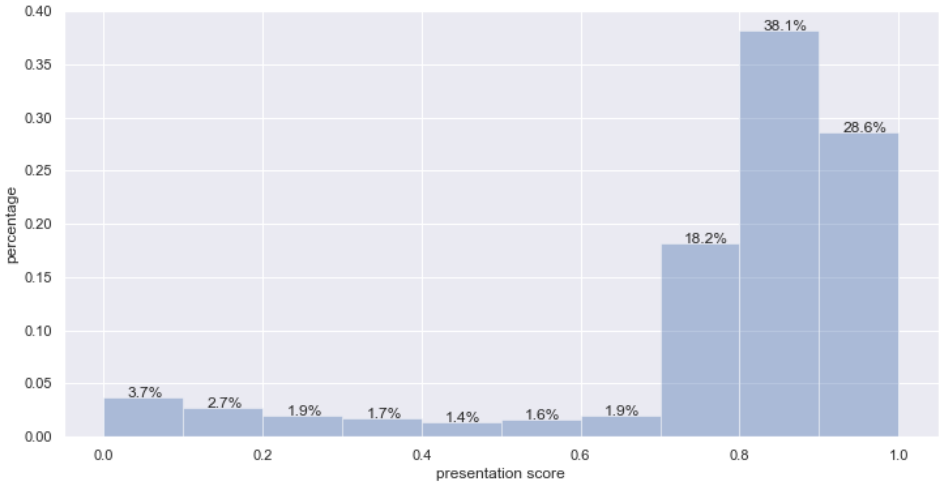




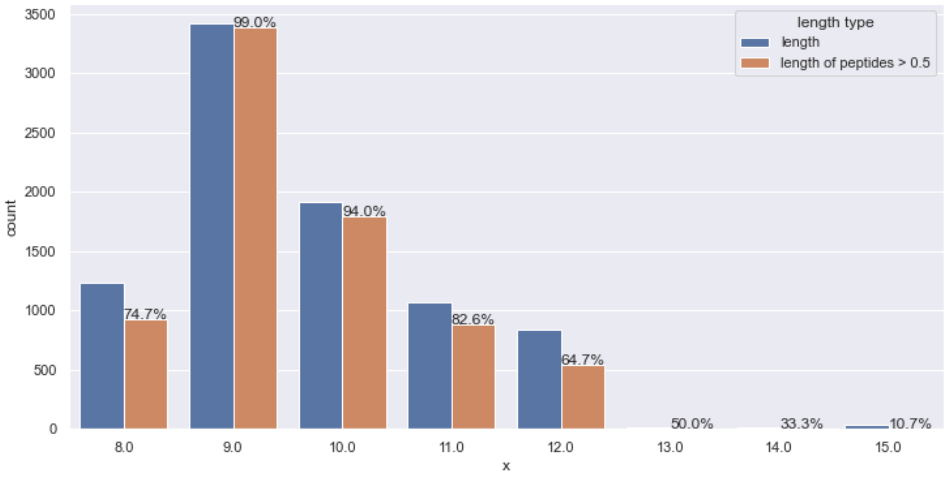




5. Reward analysis

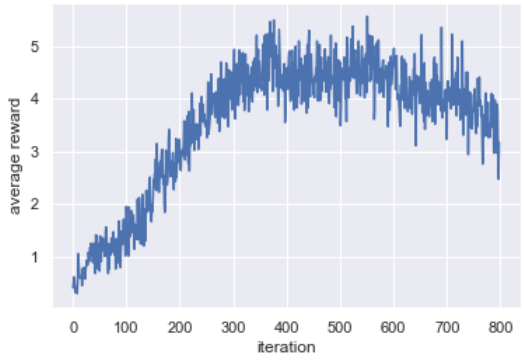


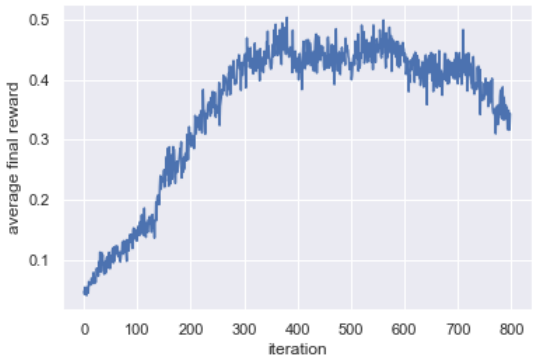
6. Length analysis



Experiment: stop criteria = 1.0

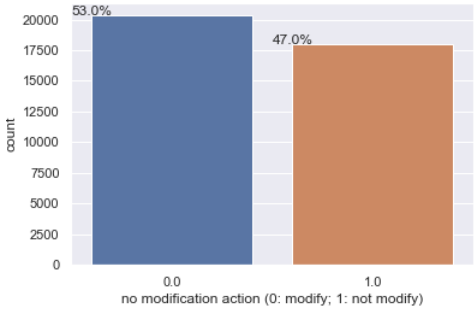
Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

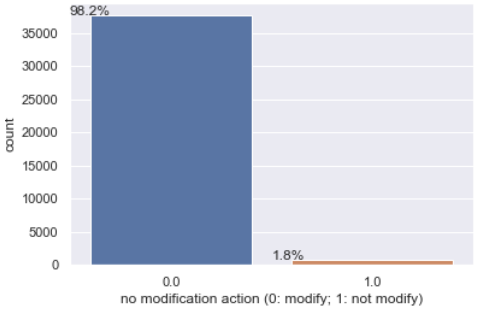


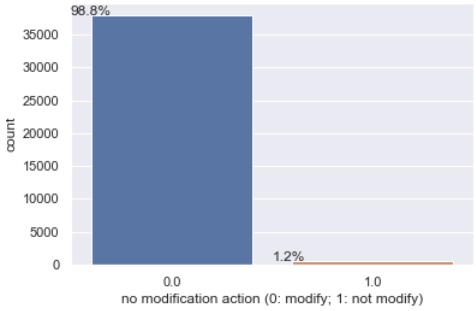


1. "No modification" action analysis

(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

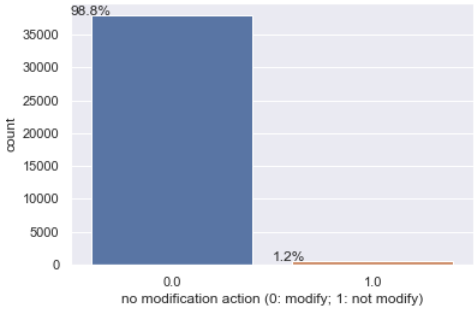


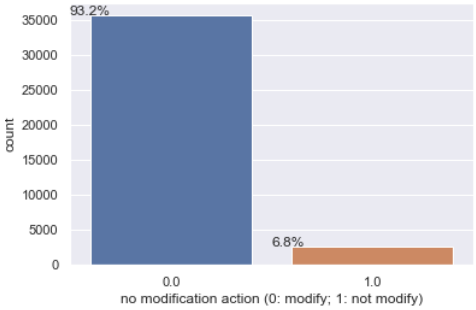




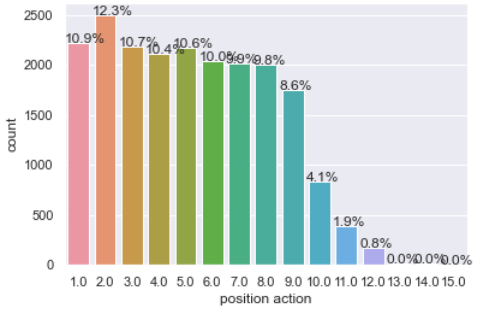
(10 iterations at 3/4)

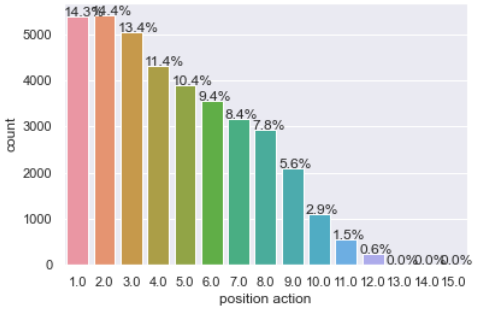
(last 10 iterations)

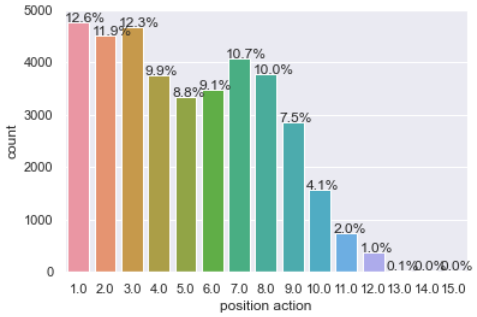


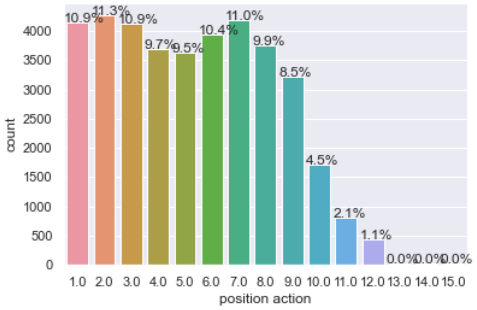


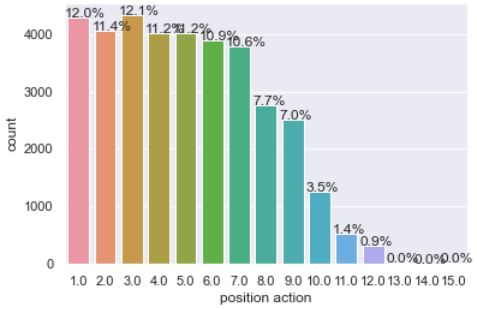
2. Position action analysis



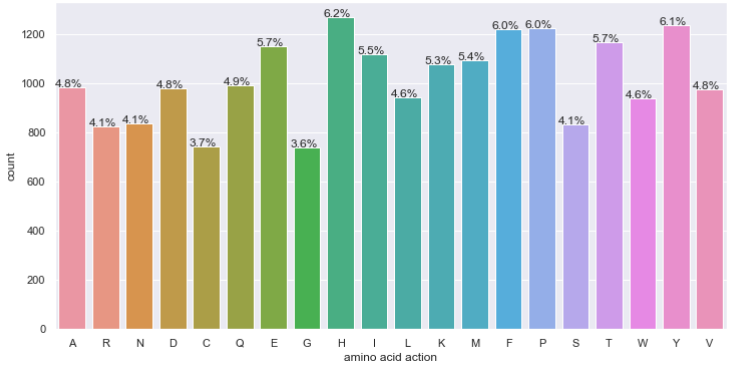


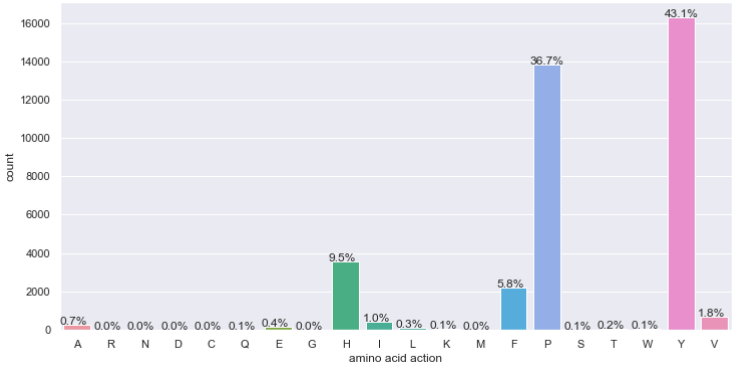


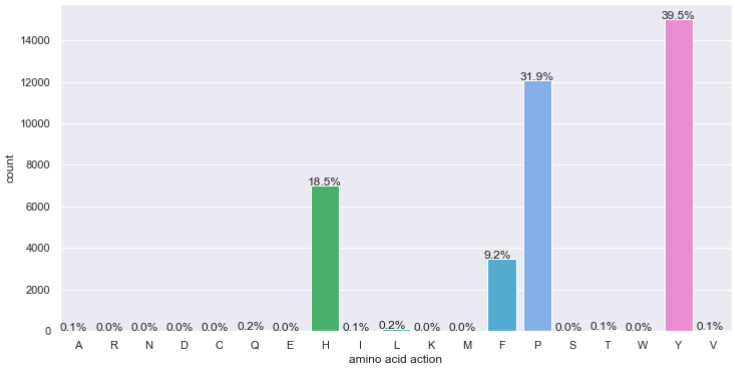


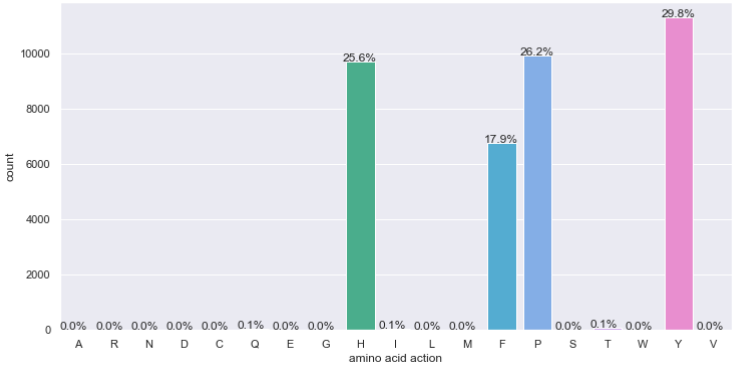


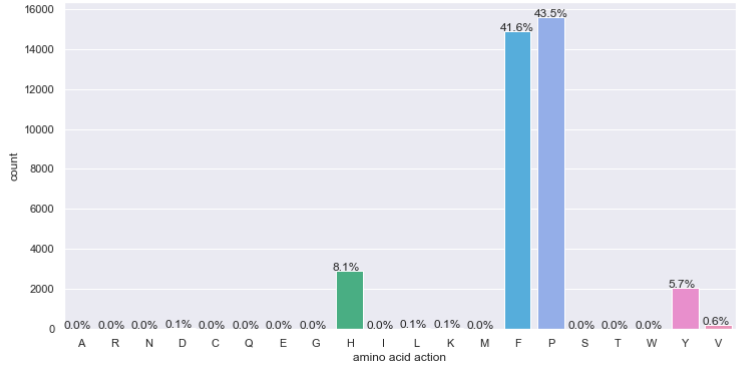
3. Amino action analysis



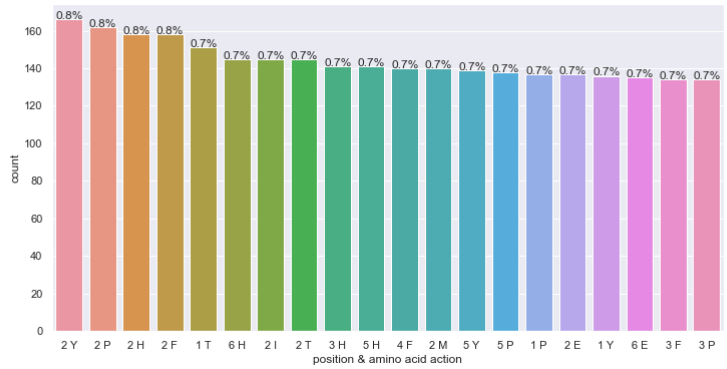


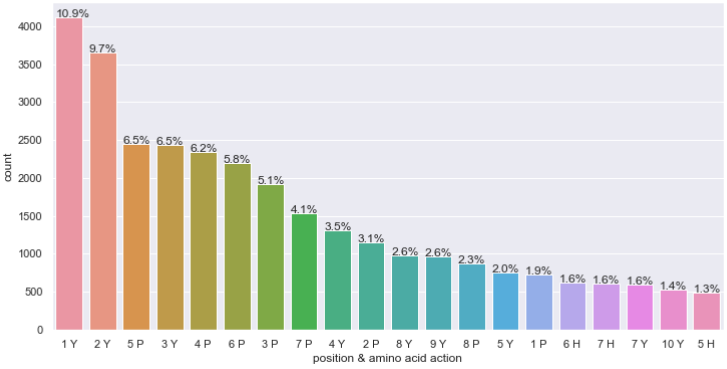


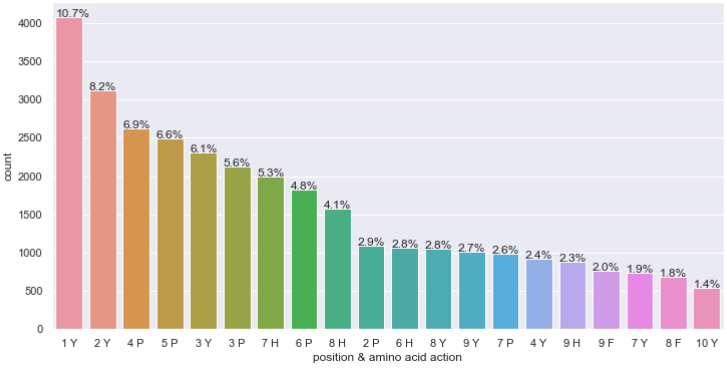


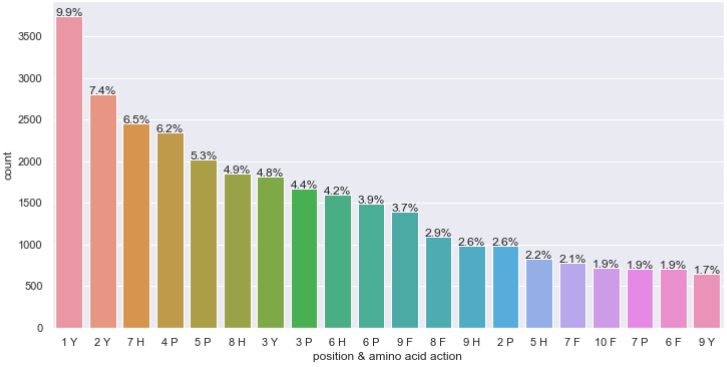


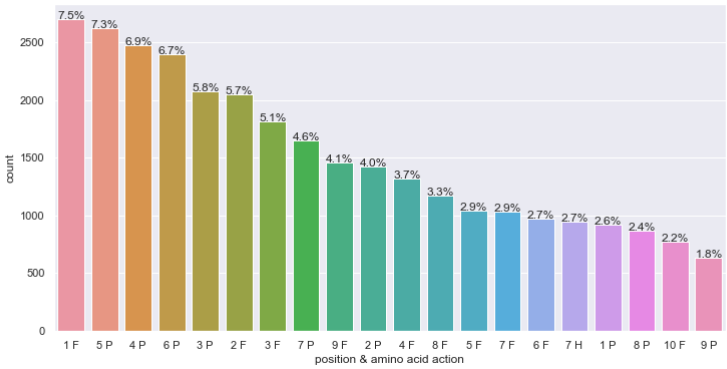
4. Position & Amino action analysis


count
position & amino acid action
0.8% 0.8% 0.8% 0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7%
2 Y 2 P 2 H 2 F 1 T 6 H 2 I 2 T 3 H 5 H 4 F 2 M 5 Y 5 P 1 P 2 E 1 Y 6 E 3 F 3 P

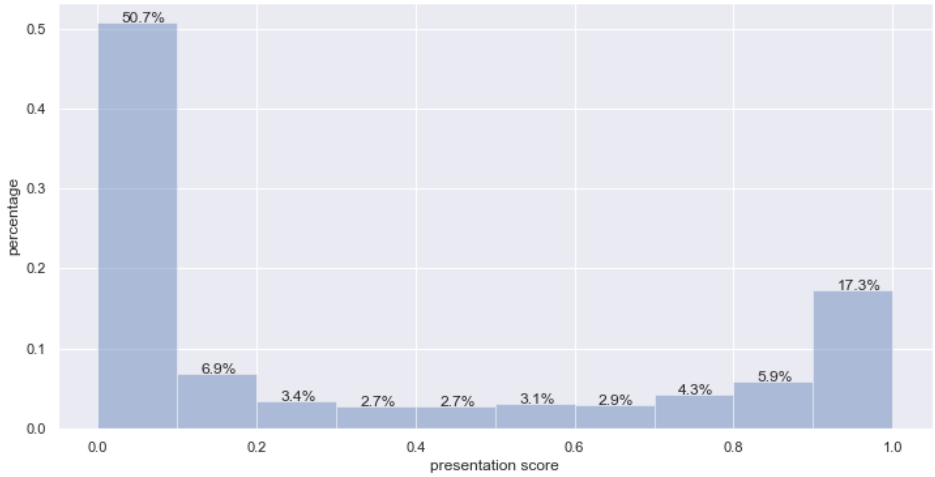

count
position & amino acid action
10.9% 9.7% 6.5% 6.5% 6.2% 5.8% 5.1% 4.1% 3.5% 3.1% 2.6% 2.6% 2.3% 2.0% 1.9% 1.6% 1.6% 1.6% 1.4% 1.3%
1 Y 2 Y 5 P 3 Y 4 P 6 P 3 P 7 P 4 Y 2 P 8 Y 9 Y 8 P 5 Y 1 P 6 H 7 H 7 Y 10 Y 5 H


count
position & amino acid action
10.7% 8.2% 6.9% 6.6% 6.1% 5.6% 5.3% 4.8% 4.1% 2.9% 2.8% 2.8% 2.7% 2.6% 2.4% 2.3% 2.0% 1.9% 1.8% 1.4%
1 Y 2 Y 4 P 5 P 3 Y 3 P 7 H 6 P 8 H 2 P 6 H 8 Y 9 Y 7 P 4 Y 9 H 9 F 7 Y 8 F 10 Y

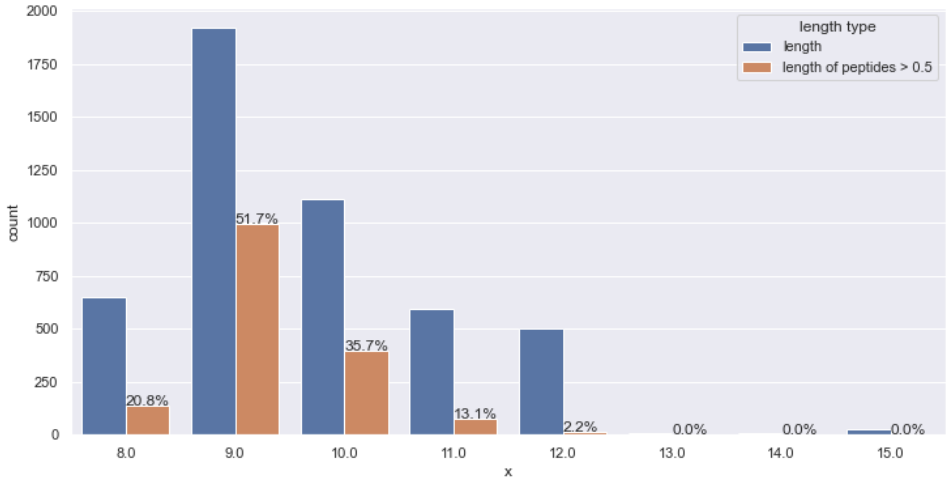

count
position & amino acid action
9.9% 7.4% 6.5% 6.2% 5.3% 4.9% 4.8% 4.4% 4.2% 3.9% 3.7% 2.9% 2.6% 2.6% 2.2% 2.1% 1.9% 1.9% 1.9% 1.7%
1 Y 2 Y 7 H 4 P 5 P 8 H 3 Y 3 P 6 H 6 P 9 F 8 F 9 H 2 P 5 H 7 F 10 F 7 P 6 F 9 Y



5. Reward analysis



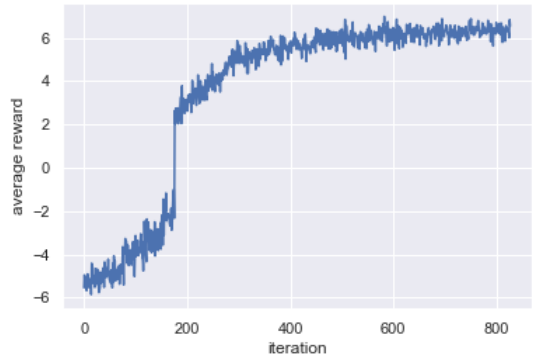
6. Length analysis

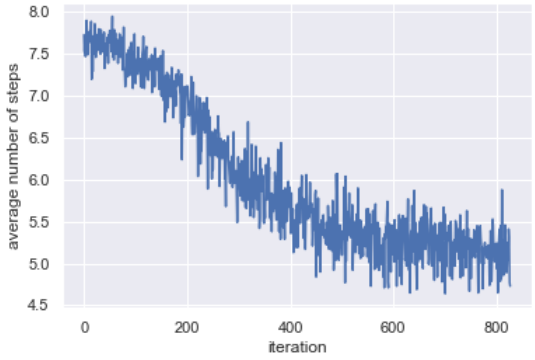


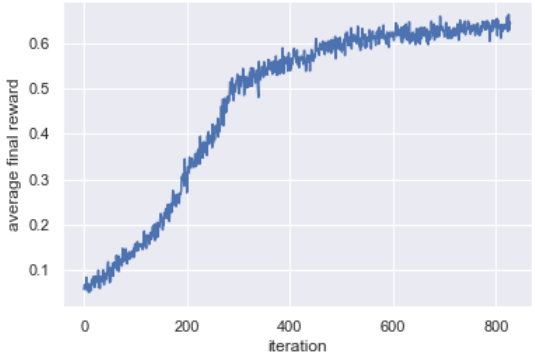
3. Result Analysis with intermediate reward at first and then only final rewards

Experiment: stop criteria = 0.6

Setting 1 (sample rate = 50% from IEDB and 50% from random)

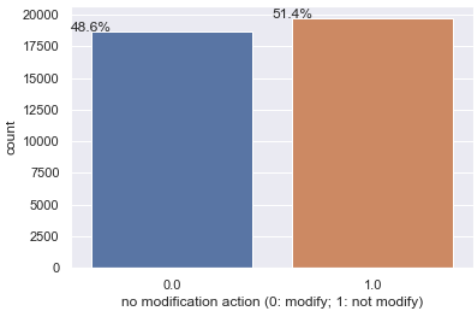


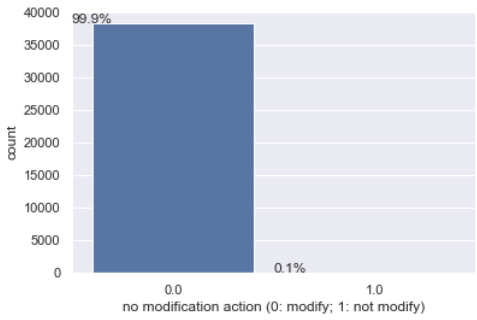


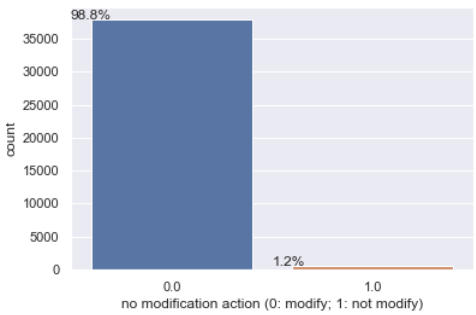


1. "No modification" action analysis

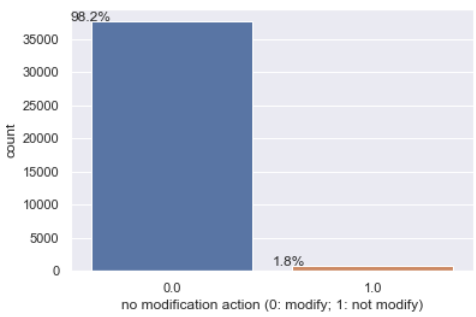
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

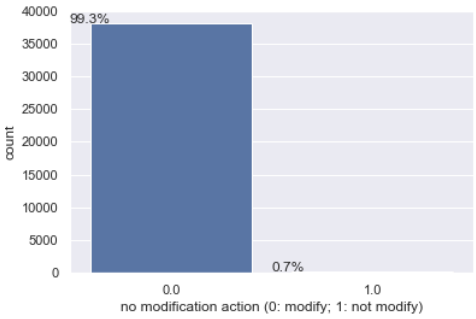




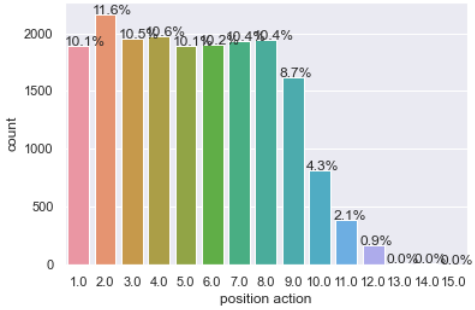


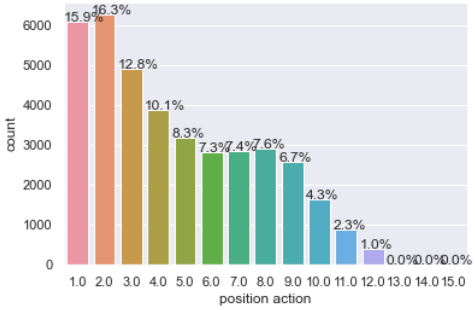
(10 iterations at 3/4) (last 10 iterations)

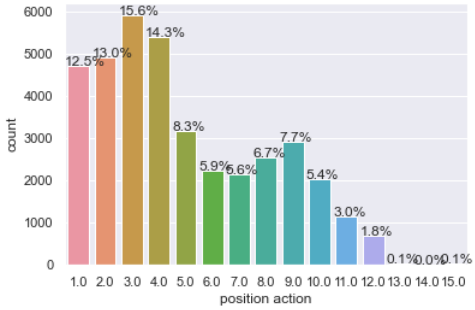


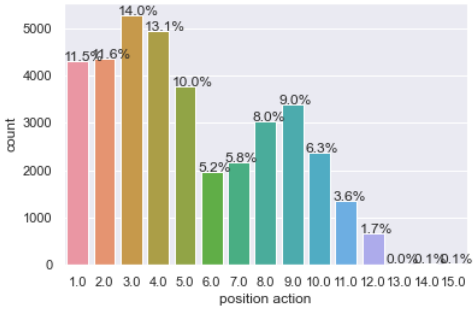


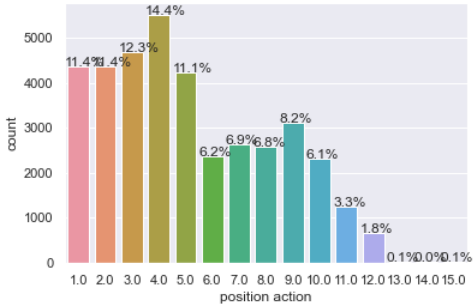
2. Position action analysis



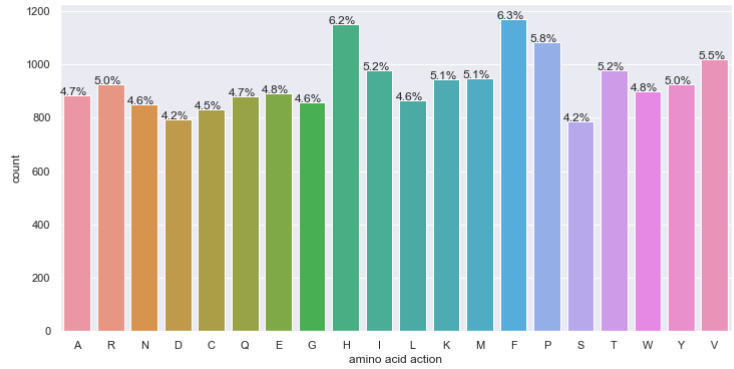


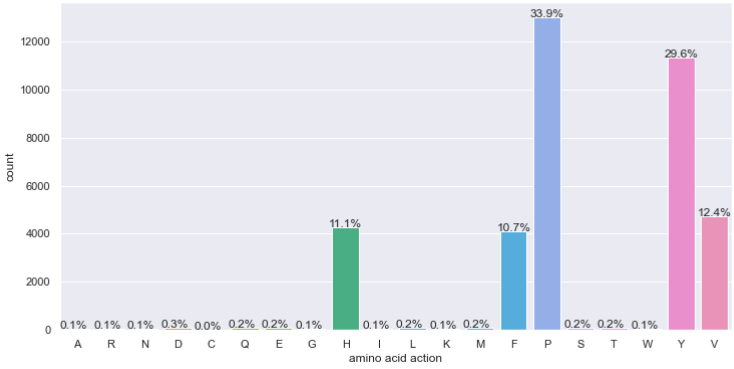


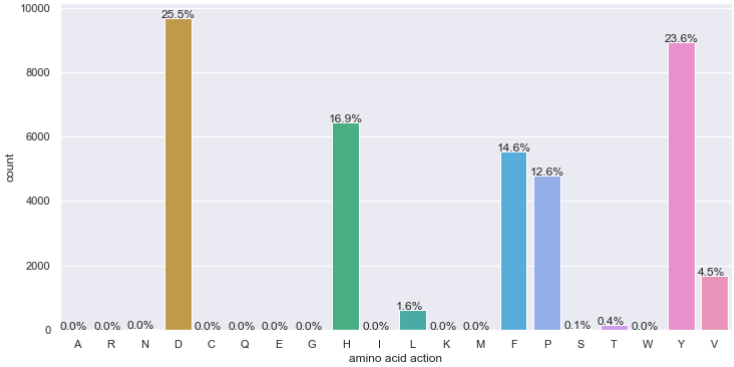


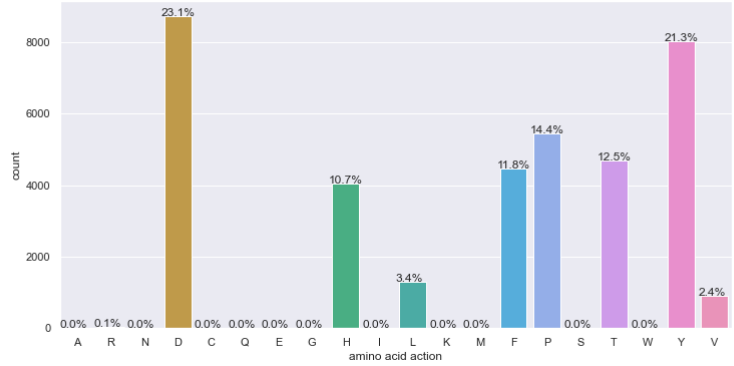


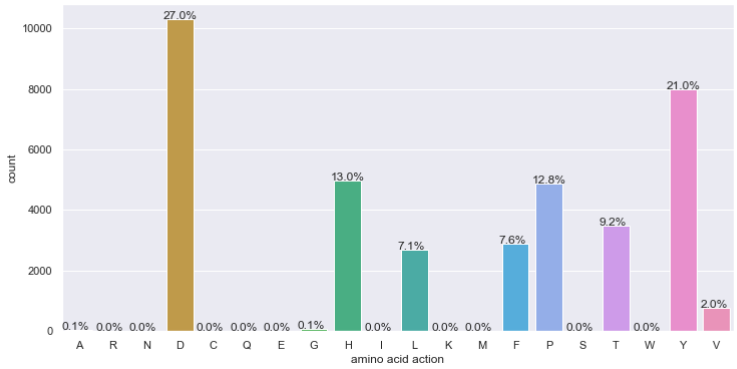
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
4.7% 5.0% 4.6% 4.2% 4.5% 4.7% 4.8% 4.6% 6.2% 5.2% 4.6% 5.1% 5.1% 6.3% 5.8% 4.2% 5.2% 4.8% 5.0% 5.5%

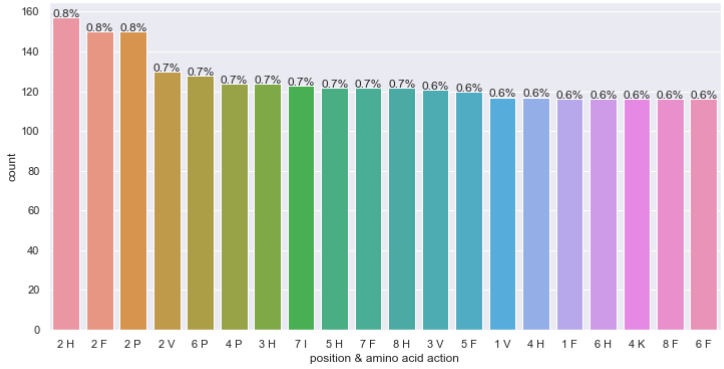

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.1% 0.1% 0.1% 0.3% 0.0% 0.2% 0.2% 0.1% 11.1% 0.1% 0.2% 0.1% 0.2% 10.7% 33.9% 0.2% 0.2% 0.1% 29.6% 12.4%

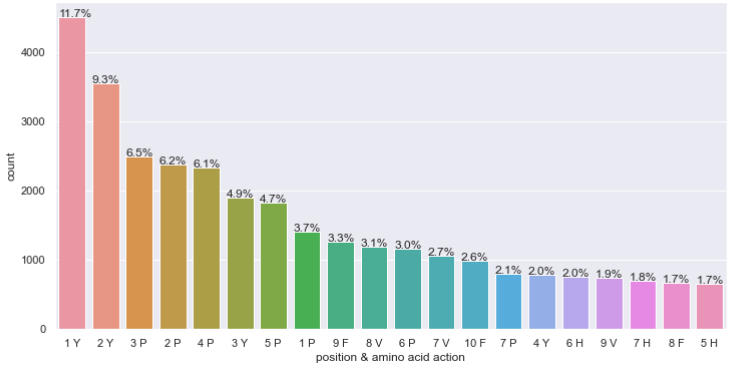

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.0% 0.0% 25.5% 0.0% 0.0% 0.0% 0.0% 16.9% 0.0% 1.6% 0.0% 0.0% 14.6% 12.6% 0.1% 0.4% 0.0% 23.6% 4.5%

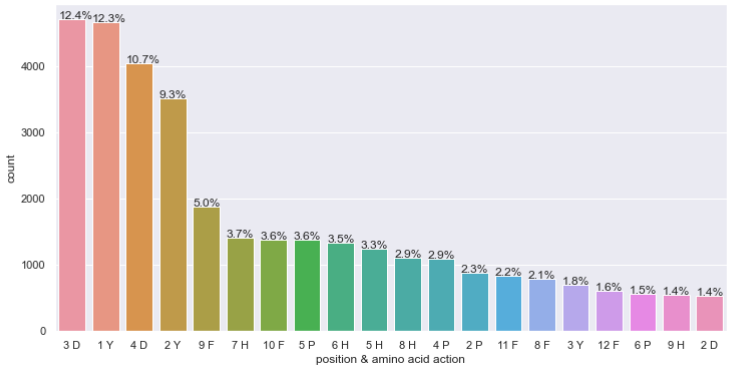

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.1% 0.0% 23.1% 0.0% 0.0% 0.0% 0.0% 10.7% 0.0% 3.4% 0.0% 0.0% 11.8% 14.4% 0.0% 12.5% 0.0% 21.3% 2.4%

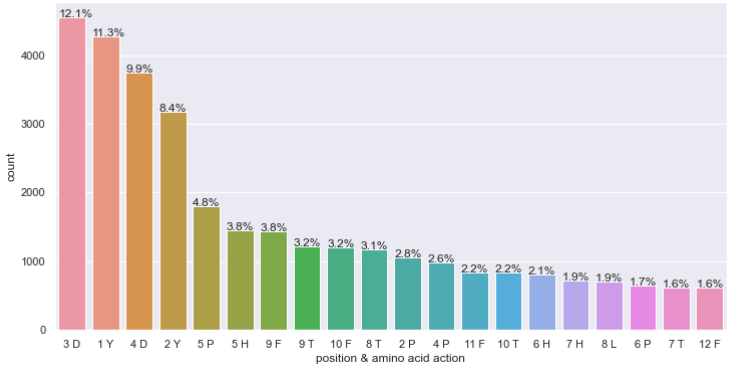


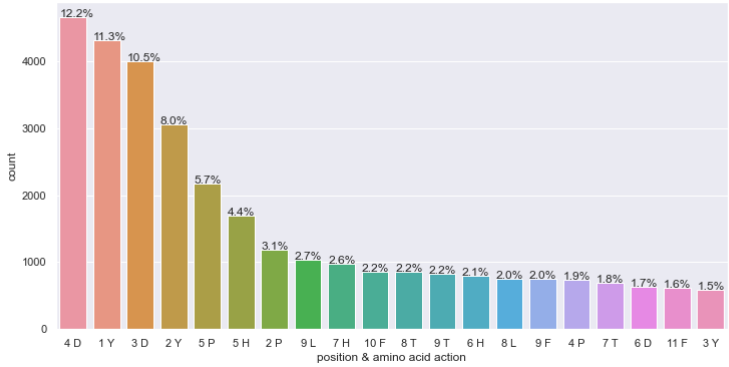
4. Position & Amino action analysis





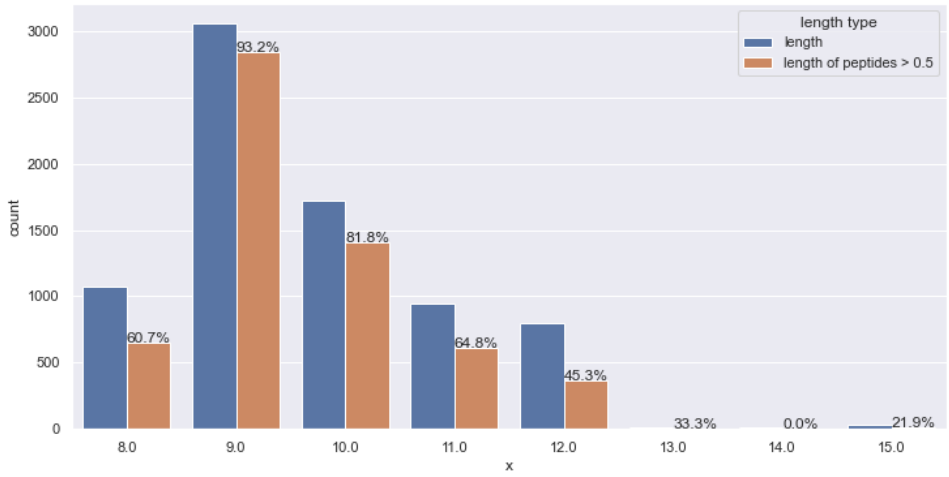






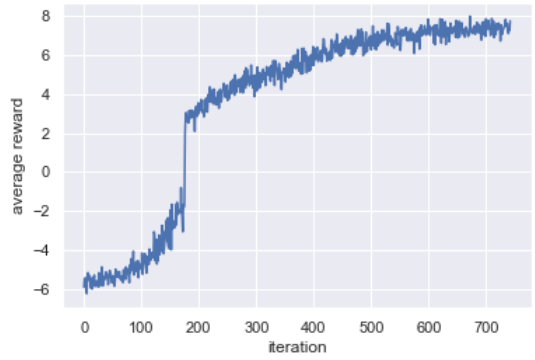
5. Reward analysis

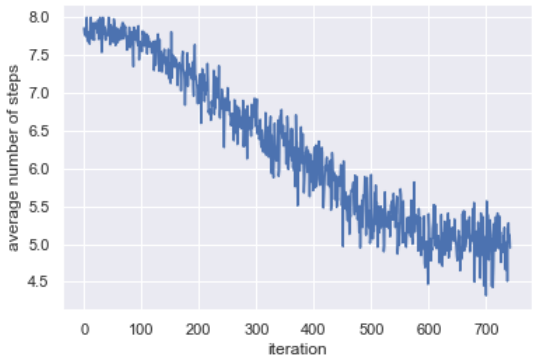
6. Length analysis

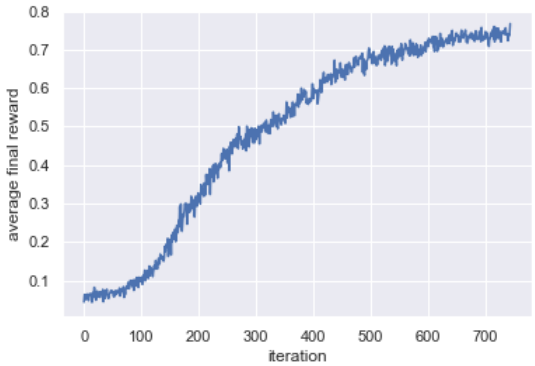


Experiment: stop criteria = 0.75

Setting 1 (sample rate = 50% from IEDB and 50% from random)

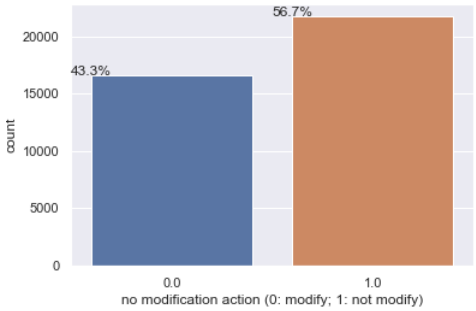


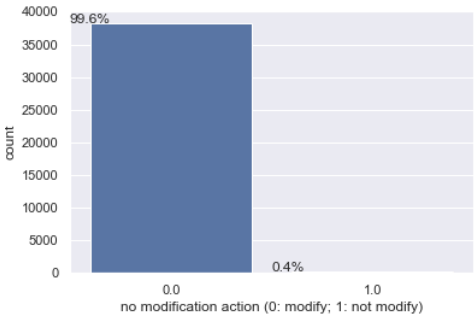


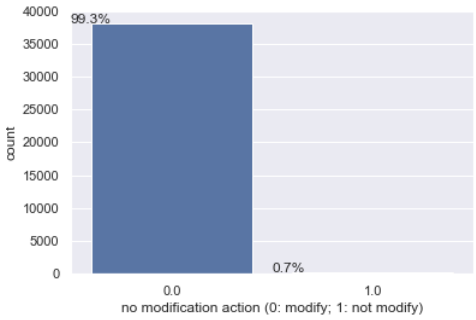


1. "No modification" action analysis

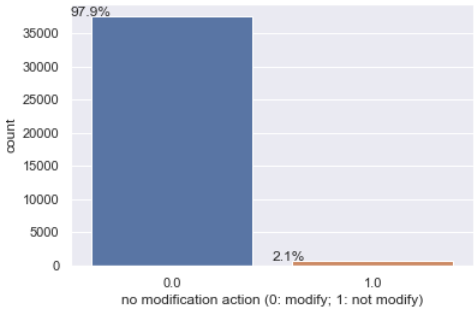
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

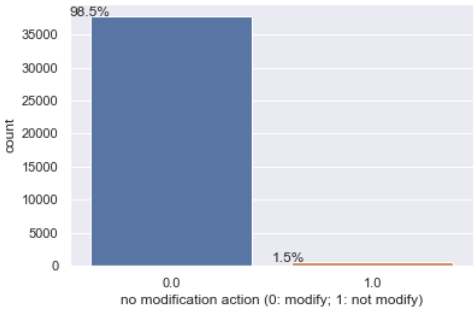




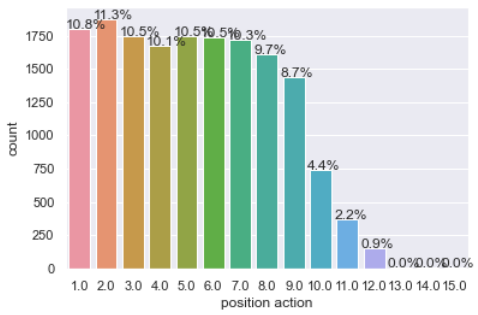


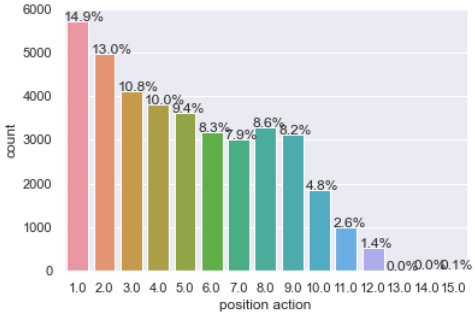
(10 iterations at 3/4) (last 10 iterations)

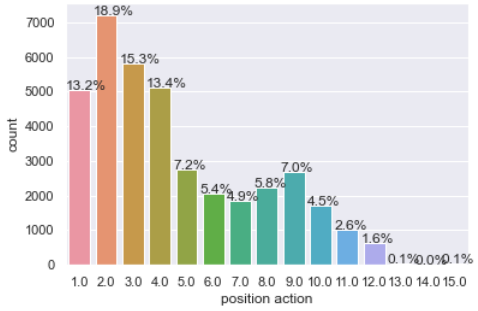


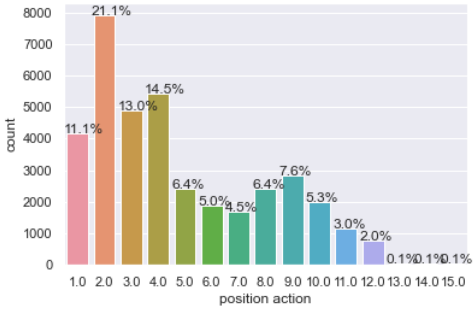


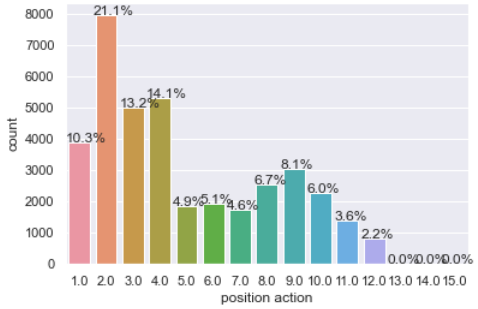
2. Position action analysis



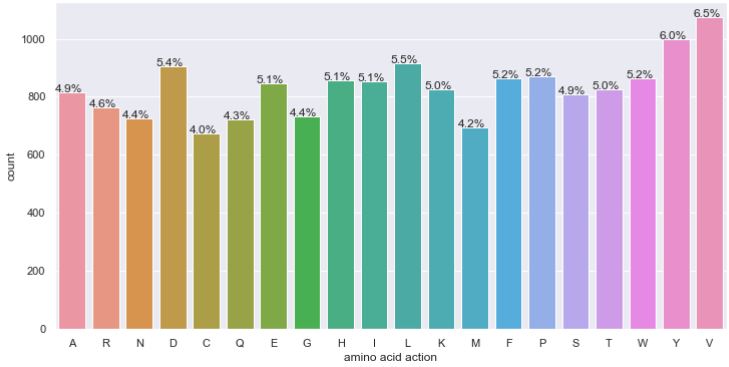


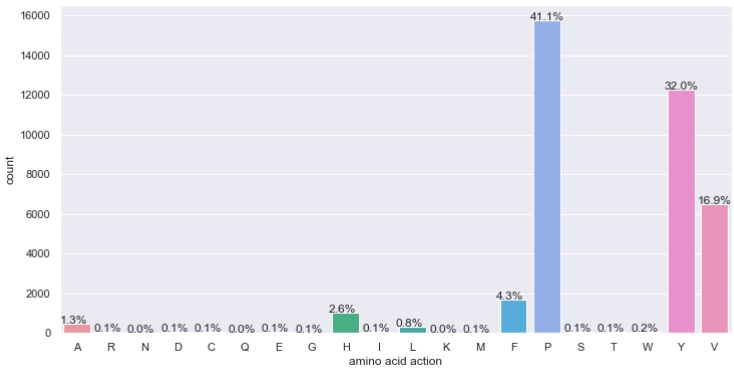


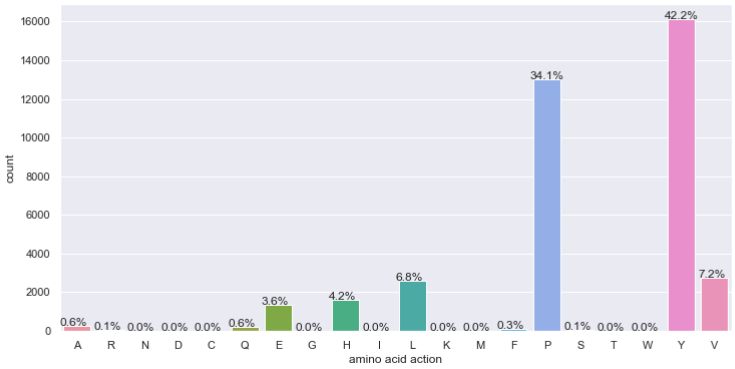


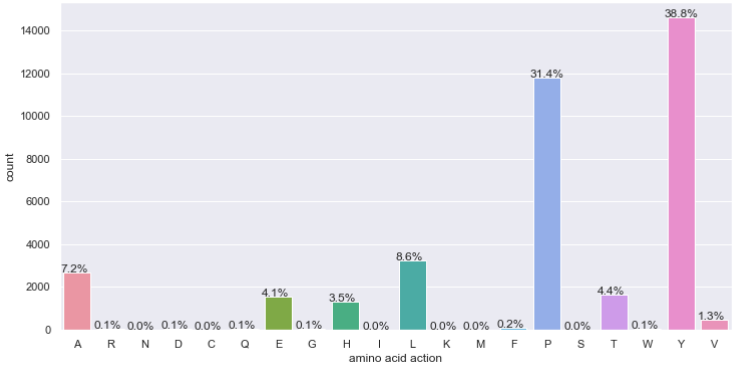


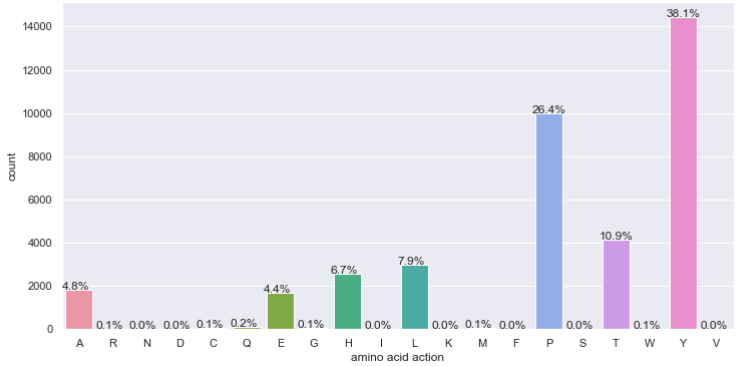
3. Amino action analysis



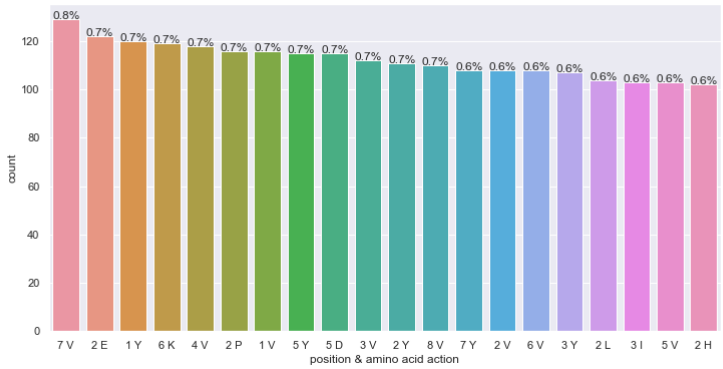


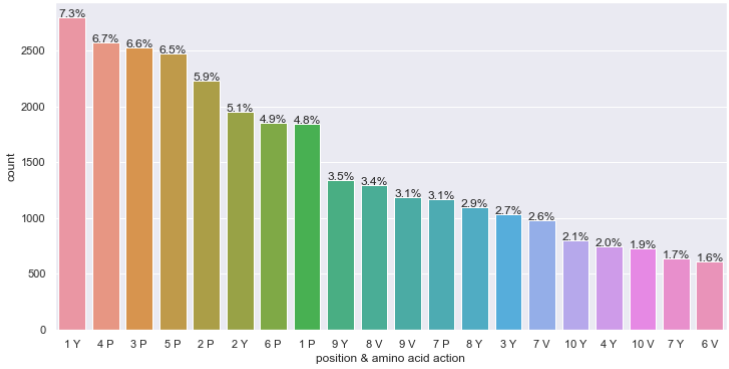


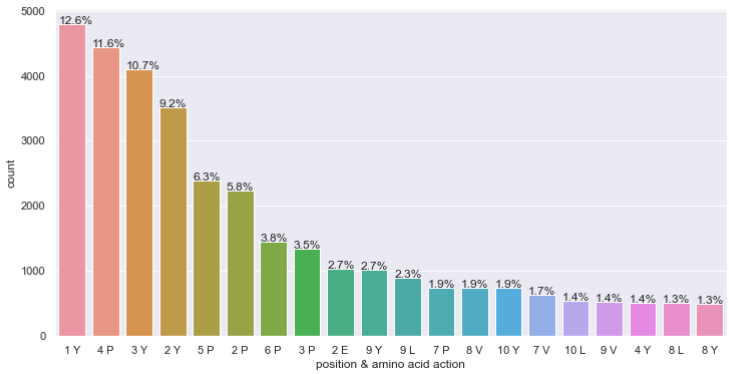


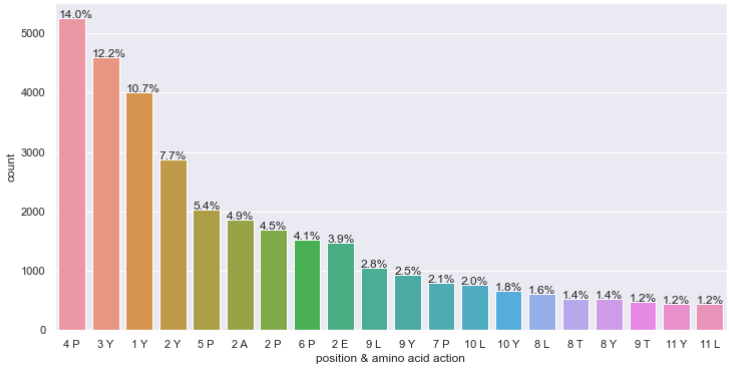


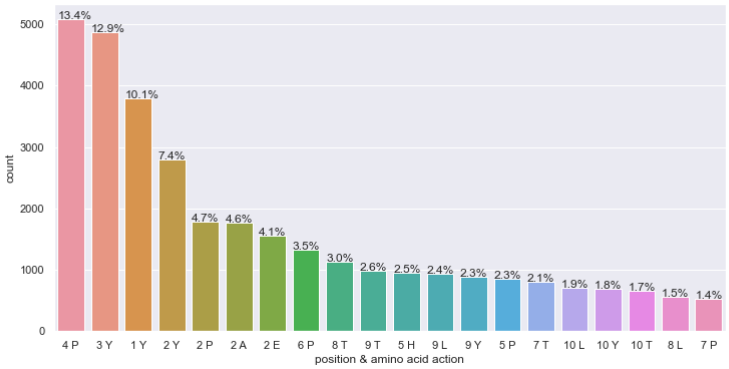
4. Position & Amino action analysis


0.8%
0.7%
0.7%
0.7%
0.7%
0.7%
0.7%
0.7%
0.7%
0.7%
0.7%
0.7%
0.6%
0.6%
0.6%
0.6%
0.6%
0.6%
0.6%
0.6%
count
0
20
40
60
80
100
120
7 V
2 E
1 Y
6 K
4 V
2 P
1 V
5 Y
5 D
3 V
2 Y
8 V
7 Y
2 V
6 V
3 Y
2 L
3 I
5 V
2 H
position & amino acid action

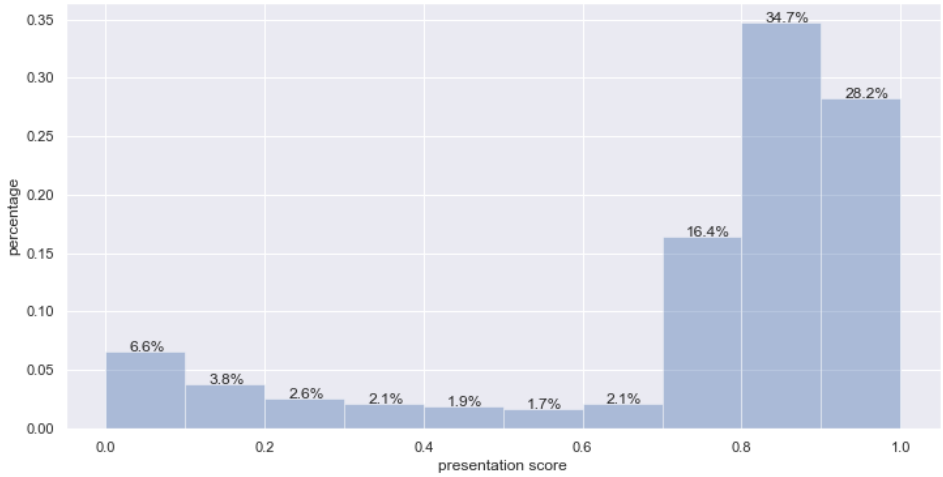

7.3%
6.7%
6.6%
6.5%
5.9%
5.1%
4.9%
4.8%
3.5%
3.4%
3.1%
3.1%
2.9%
2.7%
2.6%
2.1%
2.0%
1.9%
1.7%
1.6%
count
0
500
1000
1500
2000
2500
1 Y
4 P
3 P
5 P
2 P
2 Y
6 P
1 P
9 Y
8 V
9 V
7 P
8 Y
3 Y
7 V
10 Y
4 Y
10 V
7 Y
6 V
position & amino acid action


12.6%
11.6%
10.7%
9.2%
6.3%
5.8%
3.8%
3.5%
2.7%
2.7%
2.3%
1.9%
1.9%
1.9%
1.7%
1.4%
1.4%
1.4%
1.3%
1.3%
count
0
1000
2000
3000
4000
5000
1 Y
4 P
3 Y
2 Y
5 P
2 P
6 P
3 P
2 E
9 Y
9 L
7 P
8 V
10 Y
7 V
10 L
9 V
4 Y
8 L
8 Y
position & amino acid action

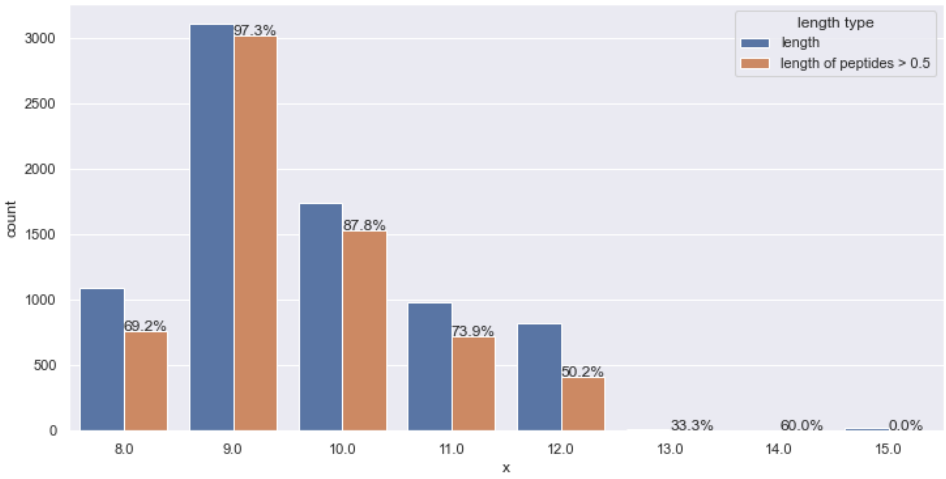

14.0%
12.2%
10.7%
7.7%
5.4%
4.9%
4.5%
4.1%
3.9%
2.8%
2.5%
2.1%
2.0%
1.8%
1.6%
1.4%
1.4%
1.2%
1.2%
1.2%
count
0
1000
2000
3000
4000
5000
4 P
3 Y
1 Y
2 Y
5 P
2 A
2 P
6 P
2 E
9 L
9 Y
7 P
10 L
10 Y
8 L
8 T
8 Y
9 T
11 Y
11 L
position & amino acid action



5. Reward analysis

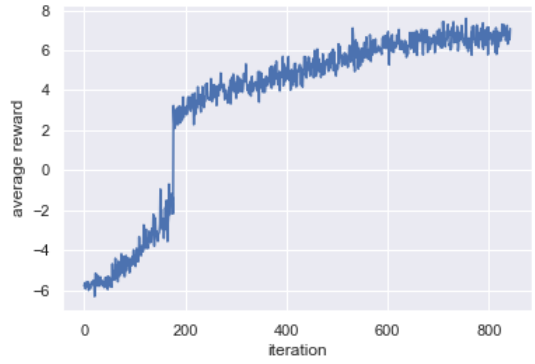


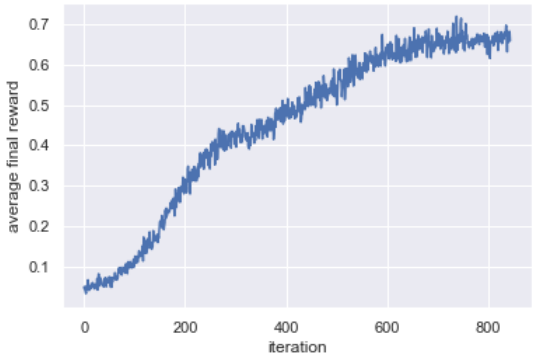
6. Length analysis



Experiment: stop criteria = 1.0

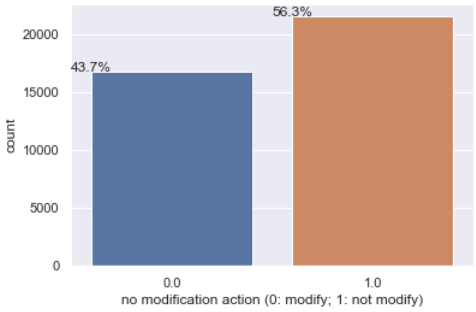
Setting 1 (sample rate = 50% from IEDB and 50% from random)

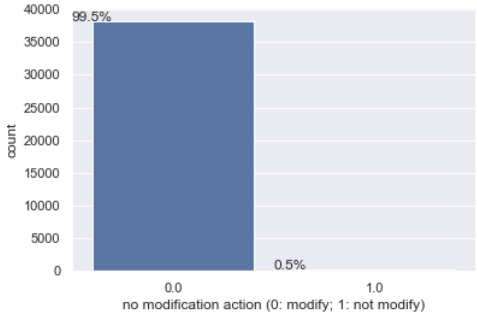


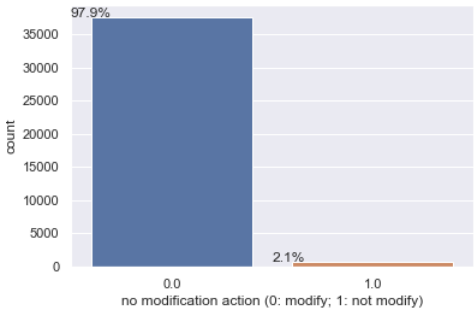


1. "No modification" action analysis

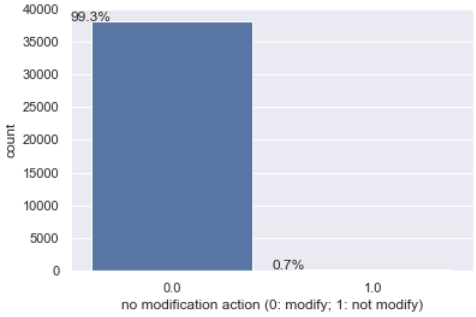
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

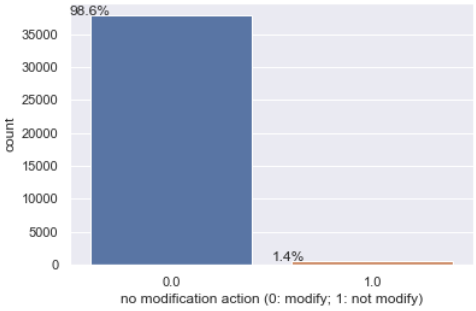




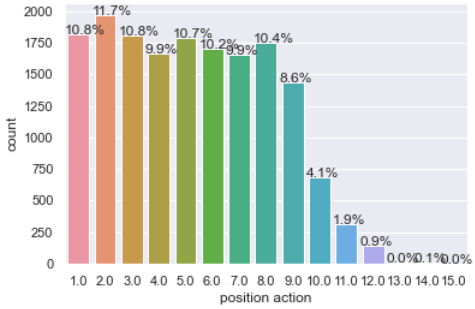


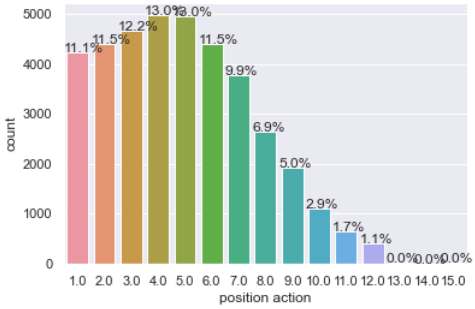
(10 iterations at 3/4) (last 10 iterations)

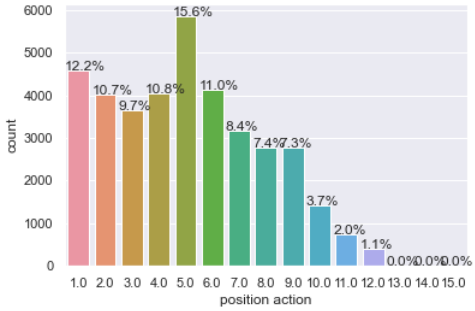


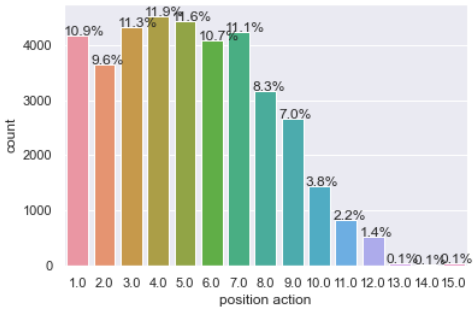


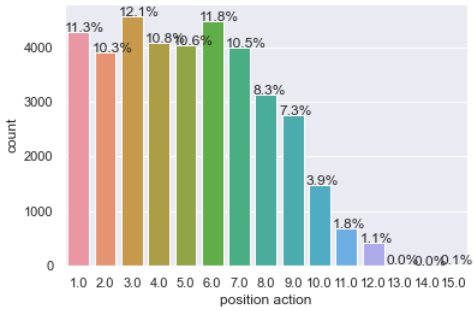
2. Position action analysis



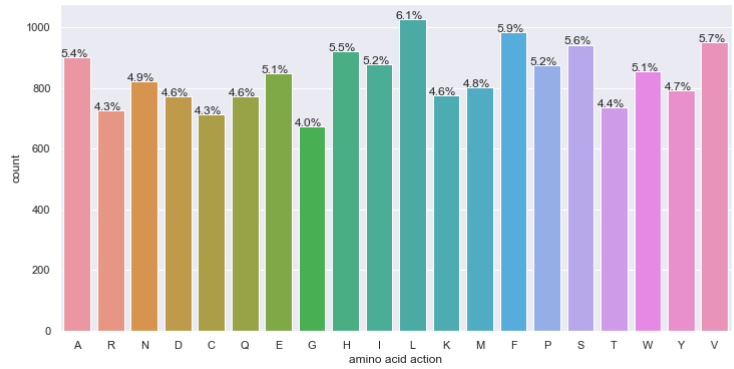


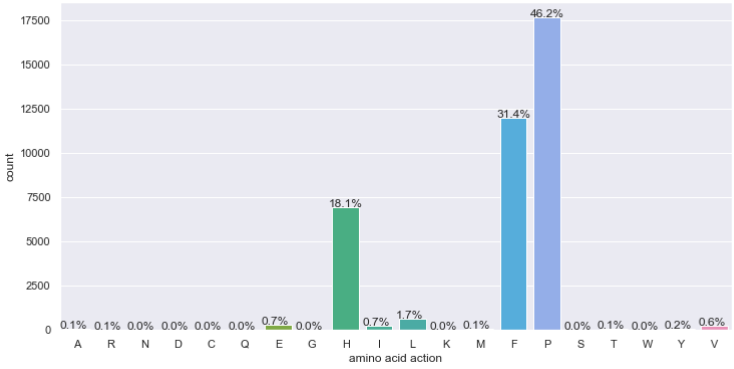


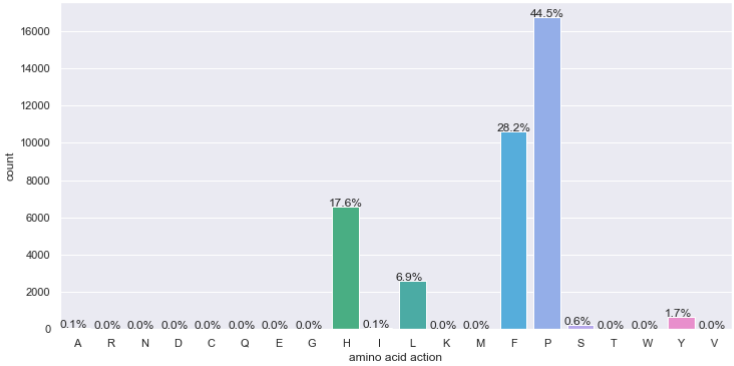


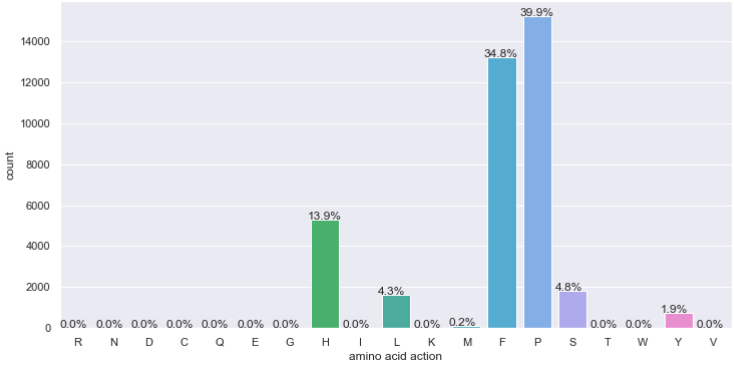


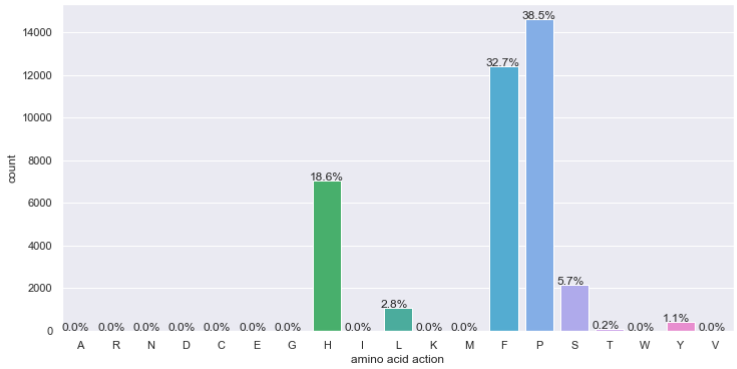
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.4% 4.3% 4.9% 4.6% 4.3% 4.6% 5.1% 4.0% 5.5% 5.2% 6.1% 4.6% 4.8% 5.9% 5.2% 5.6% 4.4% 5.1% 4.7% 5.7%

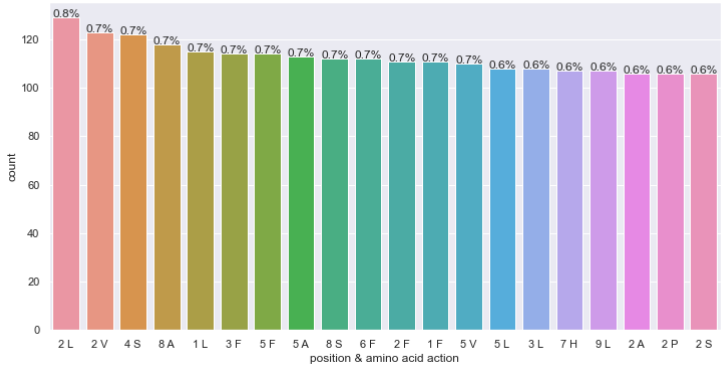

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.1% 0.1% 0.0% 0.0% 0.0% 0.0% 0.7% 0.0% 18.1% 0.7% 1.7% 0.0% 0.1% 31.4% 46.2% 0.0% 0.1% 0.0% 0.2% 0.6%

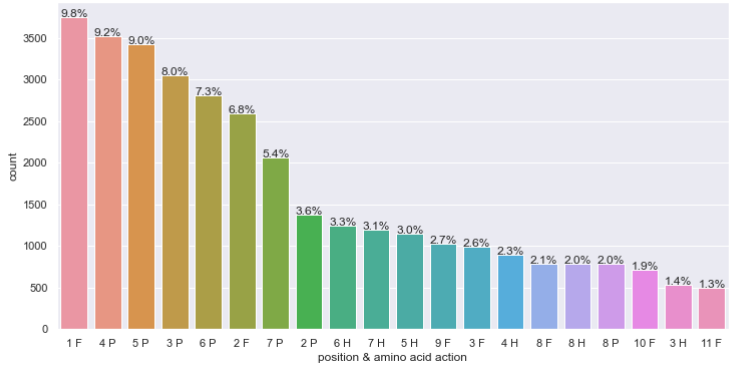

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 17.6% 0.1% 6.9% 0.0% 0.0% 28.2% 44.5% 0.6% 0.0% 0.0% 1.7% 0.0%

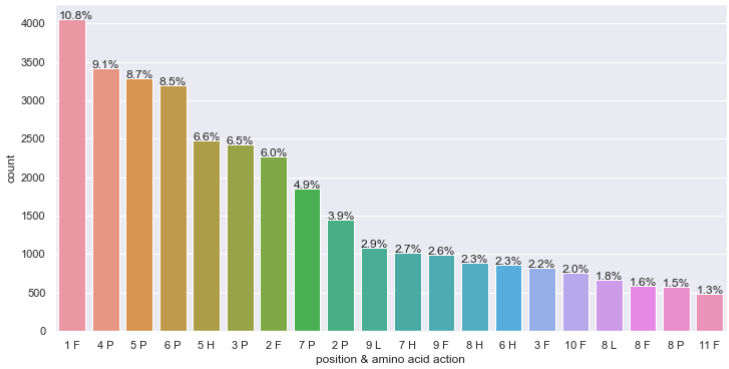

count
amino acid action
R N D C Q E G H I L K M F P S T W Y V
0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 13.9% 0.0% 4.3% 0.0% 0.2% 34.8% 39.9% 4.8% 0.0% 0.0% 1.9% 0.0%

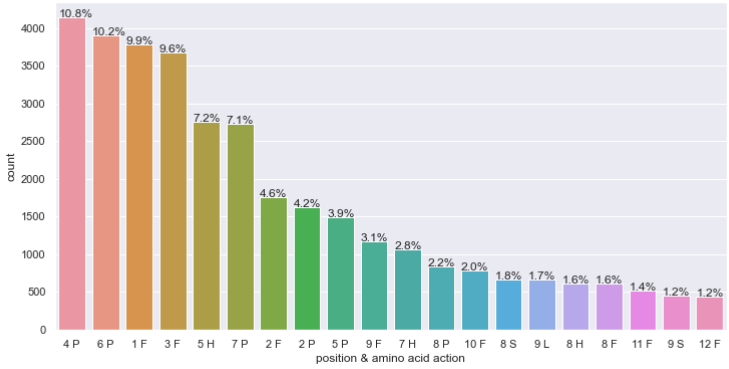


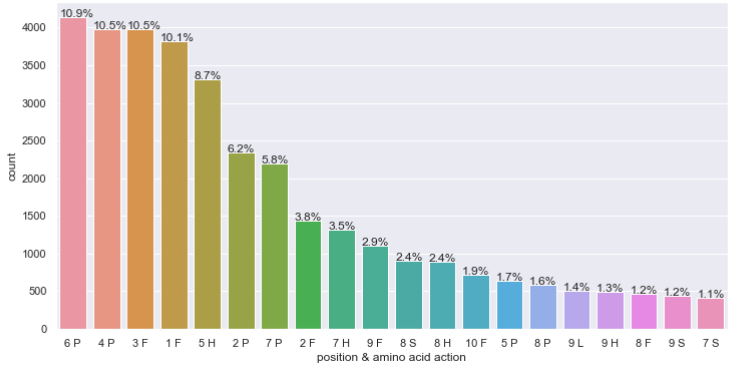
4. Position & Amino action analysis



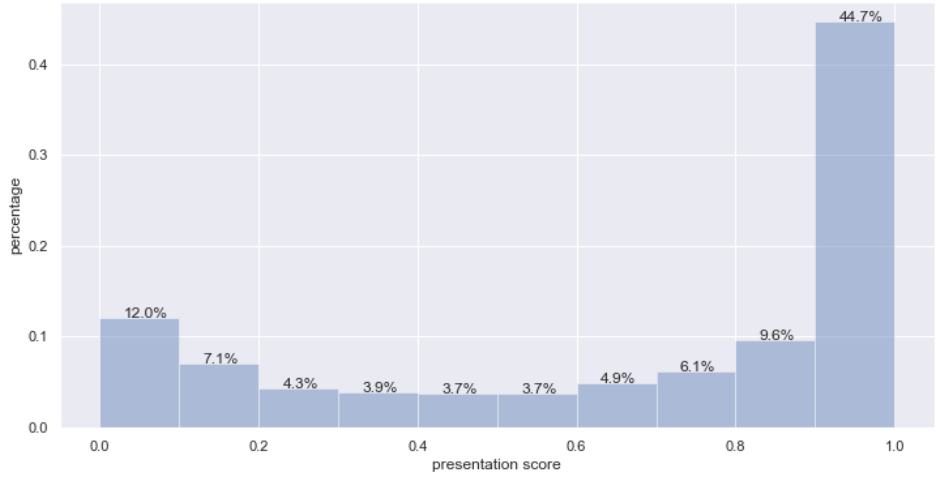




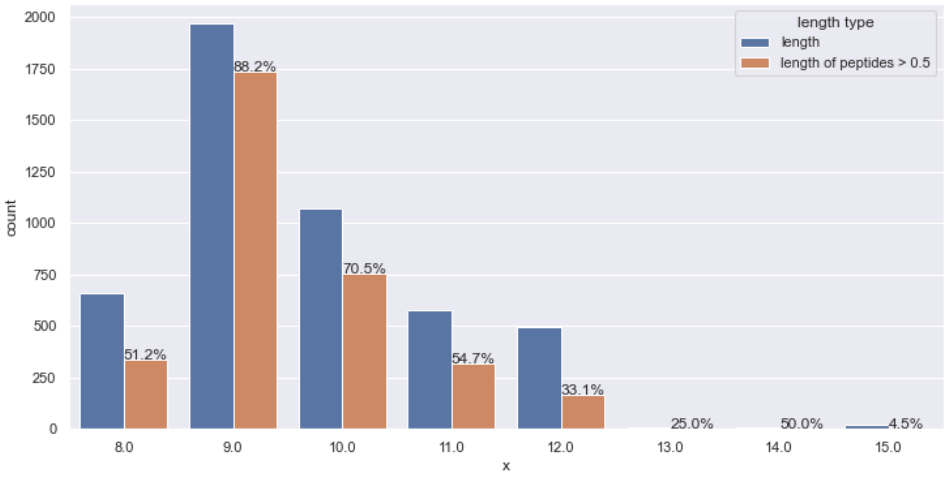




5. Reward analysis



6. Length analysis



4. Prioritized Experience Replay

In the original paper, they used Temporal Difference error to estimate the priority of each transition $(S_{t-1}, A_{t-1}, S_t, R_t)$.

TD error: $\delta_t = R_t + \gamma \max_a(Q(S_t, a)) - Q(S_{t-1}, A_{t-1})$

$Q = R_t + \gamma * (R_{t+1} + \ldots + \gamma^{T-t-1} R_T)$

Priority: $p_i = |\delta_i| + \epsilon$ ---> $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$

"The exponent α determines how much prioritization is used, with $\alpha = 0$ corresponding to the uniform case."

Correction with importance-sampling weights: $w_i = (\frac{1}{N} \cdot \frac{1}{P(i)})^\beta$

With $P(i)$, transitions are sampled with non-uniform probability. By setting $\beta = 1$ at the end of training, the above weight can fully compensates for the non-uniform probabilities. At the beginning of training, the value of β should be small.

$\max(w_i)$ is used to normalize the weight w_i to be smaller than 1.

In our experiments, instead of using TD error to calculate the priority of single transition, we can use the hamming distance and the final presentation scores to calculate the priorities of the entire episodes. The episode with smaller hamming distance and higher presentation scores will have higher priority.

How to design the priority function?

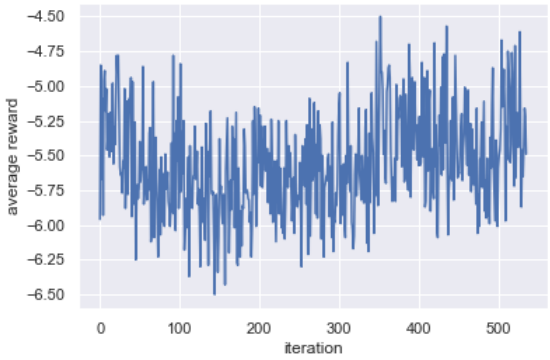
$p_i = (d_{\max} - d + 1)^\phi \cdot s_T$ ---> $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$

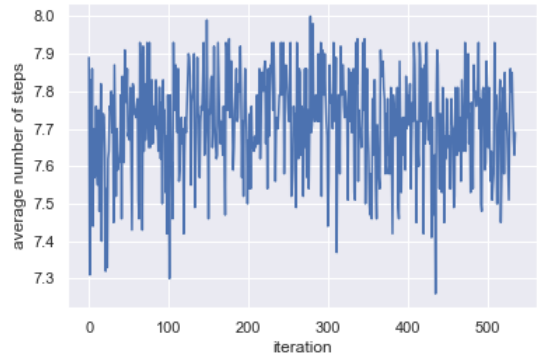
change s_T to $(s_T - s_0)$

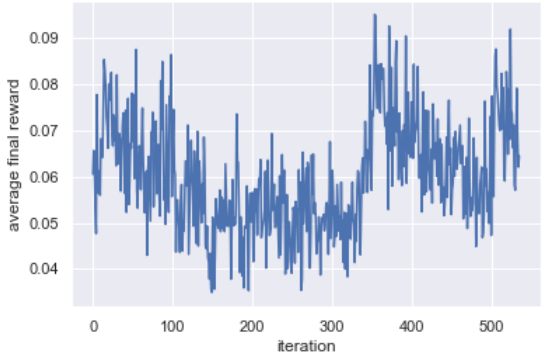
where s_T is the final presentation score; d is the hamming distance between the initial state and the termination state; $d_{\max}$ is the maximum number of steps; ϕ and α are two hyperparameters with values smaller than 1. Currently, both ϕ and α are set to be 0.5.

Note that $(d_{\max} - d + 1)$ ensures that the value is greater than 0.

- Setting 1 (stop criteria = 0.6; sample rate = 50% from IEDB and 50% from random)

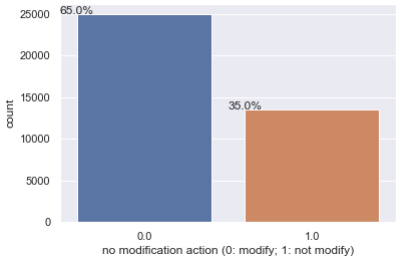


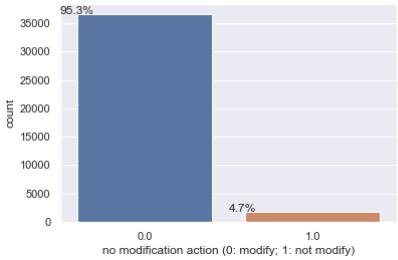


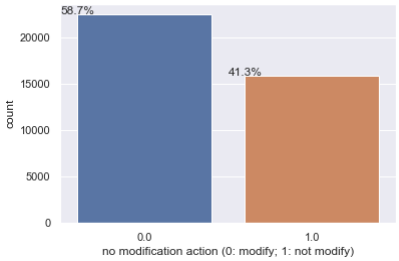


1. "No modification" action analysis

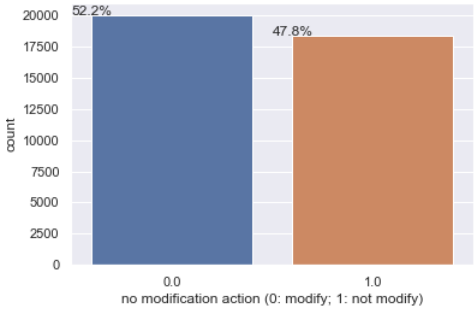
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

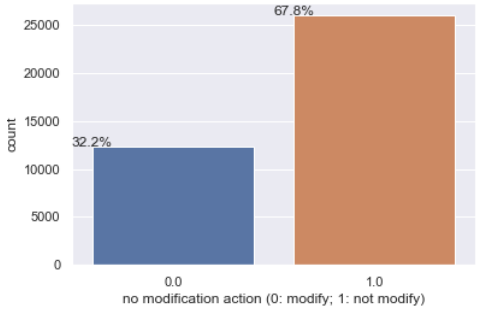




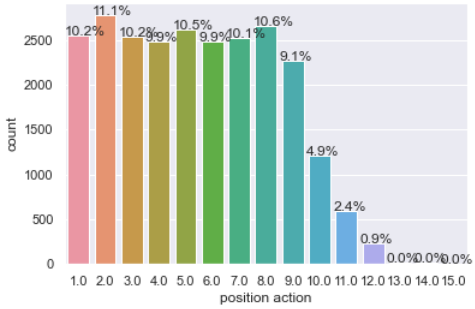


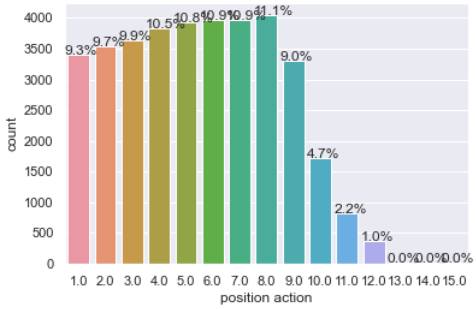
(10 iterations at 3/4) (last 10 iterations)

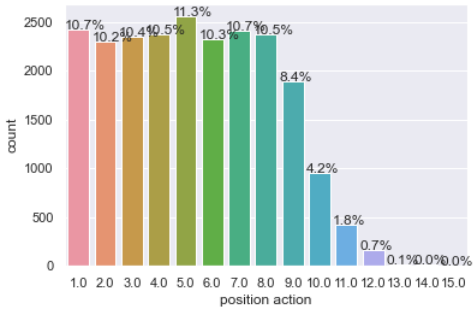


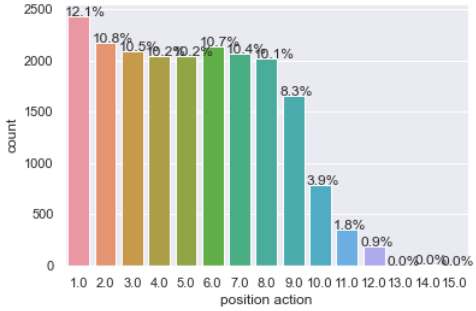


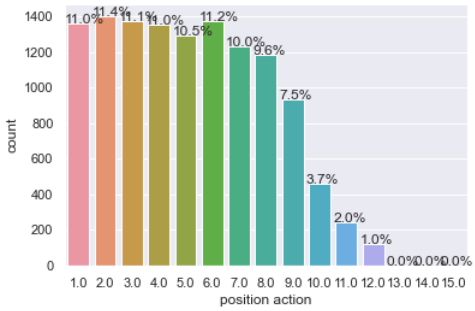
2. Position action analysis



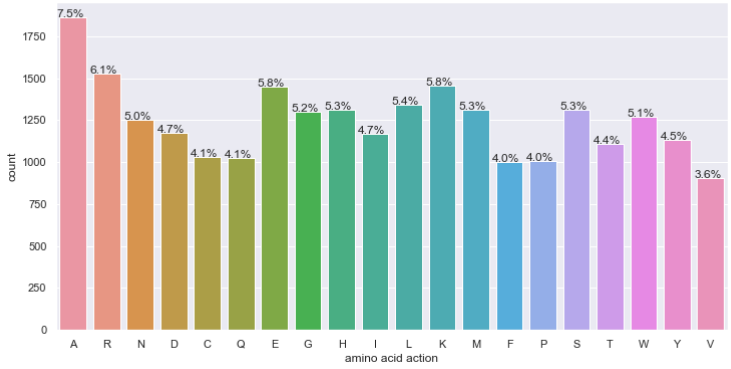


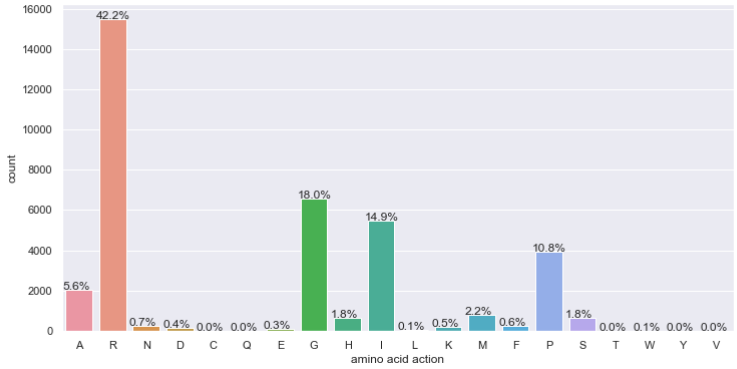


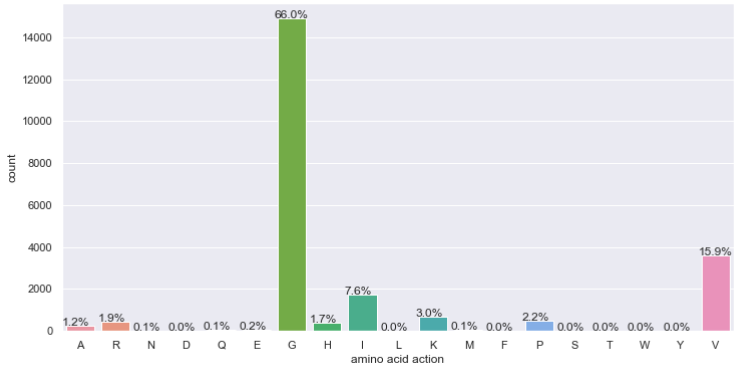


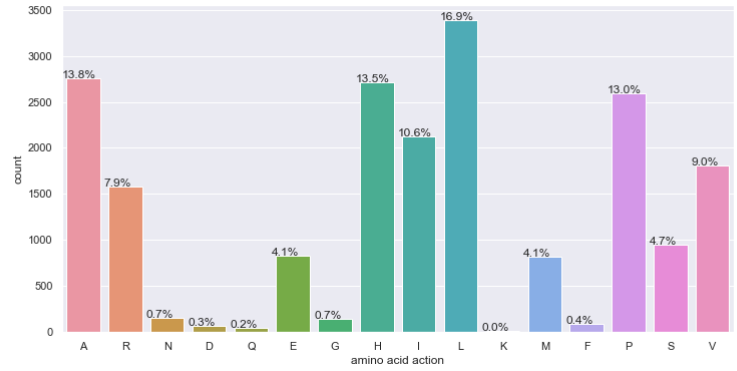


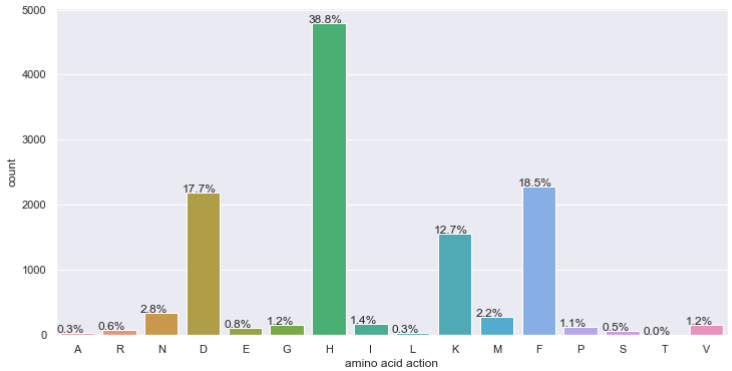
3. Amino action analysis



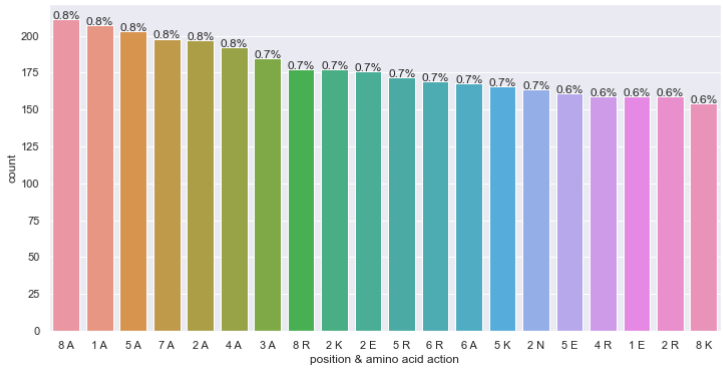


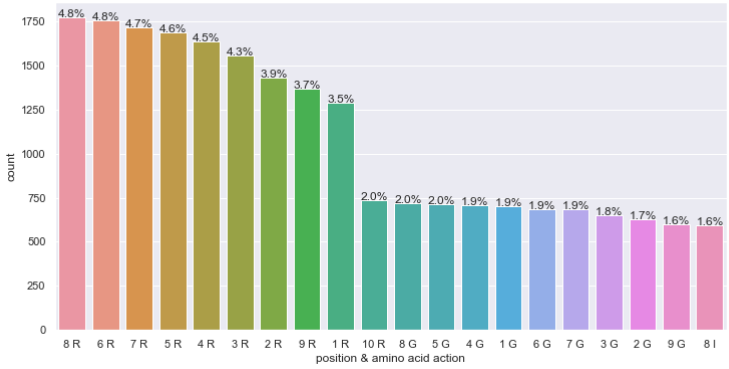


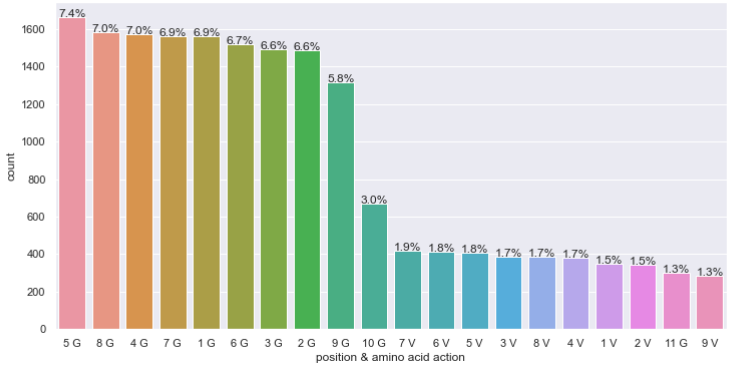


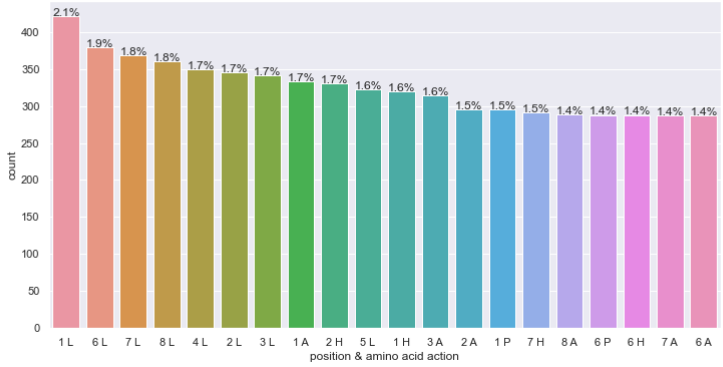


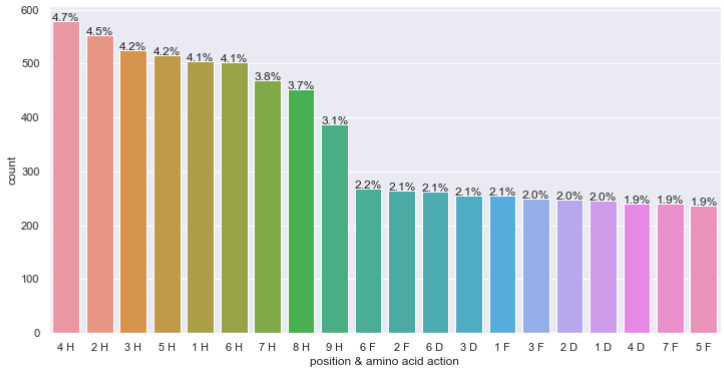
4. Position & Amino action analysis


count
0.8% 0.8% 0.8% 0.8% 0.8% 0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.6% 0.6% 0.6% 0.6% 0.6%
8 A 1 A 5 A 7 A 2 A 4 A 3 A 8 R 2 K 2 E 5 R 6 R 6 A 5 K 2 N 5 E 4 R 1 E 2 R 8 K
position & amino acid action

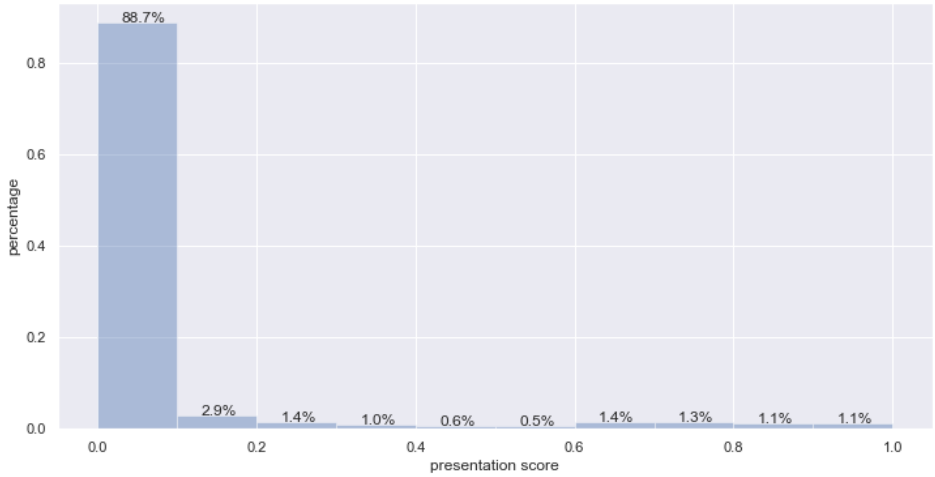

count
4.8% 4.8% 4.7% 4.6% 4.5% 4.3% 3.9% 3.7% 3.5% 2.0% 2.0% 2.0% 1.9% 1.9% 1.9% 1.9% 1.8% 1.7% 1.6% 1.6%
8 R 6 R 7 R 5 R 4 R 3 R 2 R 9 R 1 R 10 R 8 G 5 G 4 G 1 G 6 G 7 G 3 G 2 G 9 G 8 I
position & amino acid action


count
7.4% 7.0% 7.0% 6.9% 6.9% 6.7% 6.6% 6.6% 5.8% 3.0% 1.9% 1.8% 1.8% 1.7% 1.7% 1.7% 1.5% 1.5% 1.3% 1.3%
5 G 8 G 4 G 7 G 1 G 6 G 3 G 2 G 9 G 10 G 7 V 6 V 5 V 3 V 8 V 4 V 1 V 2 V 11 G 9 V
position & amino acid action

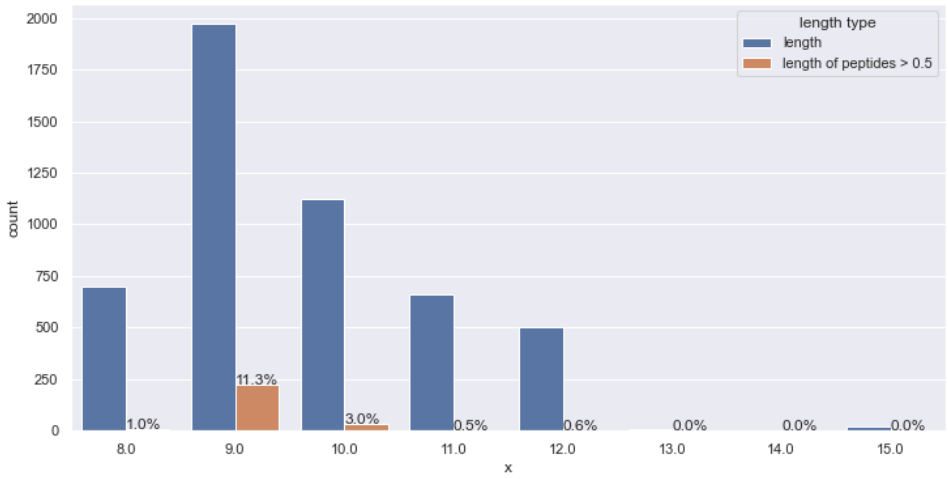

count
2.1% 1.9% 1.8% 1.8% 1.7% 1.7% 1.7% 1.7% 1.7% 1.6% 1.6% 1.6% 1.5% 1.5% 1.5% 1.4% 1.4% 1.4% 1.4% 1.4%
1 L 6 L 7 L 8 L 4 L 2 L 3 L 1 A 2 H 5 L 1 H 3 A 2 A 1 P 7 H 8 A 6 P 6 H 7 A 6 A
position & amino acid action



5. Reward analysis



6. Length analysis



5. Using undiscounted factors for "no modification" actions at the end of episodes

The returns for "modification" actions on non-terminal states S_{t-1}: $y = R_t + \gamma V(S_{t+1})$, that is the sum of current reward R_t and the estimated future rewards $V(S_{t+1})$.

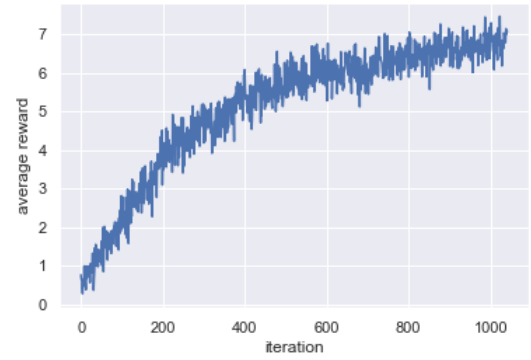
If states are terminal states, the return $y = R_T$.

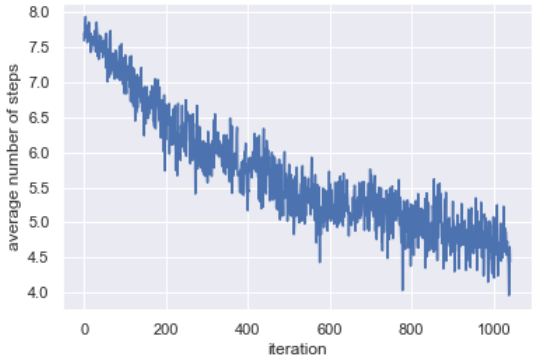
This return can be used to calculate the advantage function or TD error and update the value function.

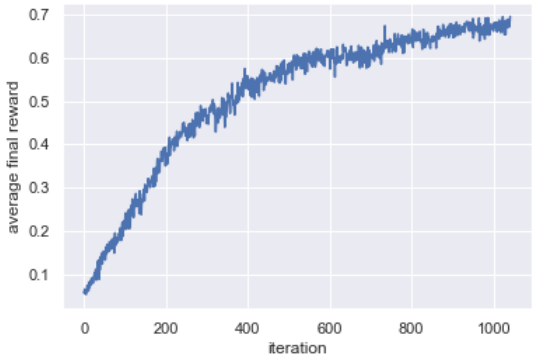
Instead of using the return function $y = R_t + \gamma V(S_{t+1})$ for those "no modification" transitions at the end of episodes, we can use $y = R_T$ to calculate their returns. In other words, all these "no modification" transitions are viewed as the same termination state.

Experiment: stop criteria = 0.6

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

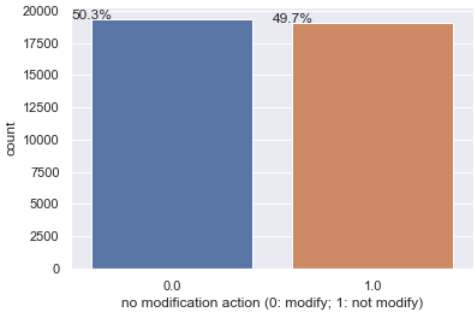
- 

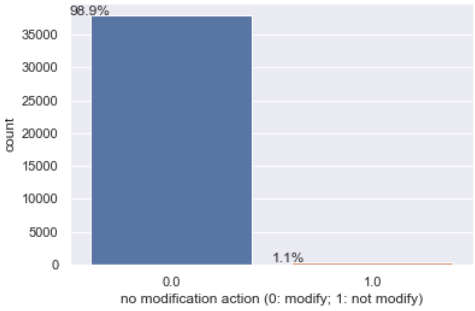


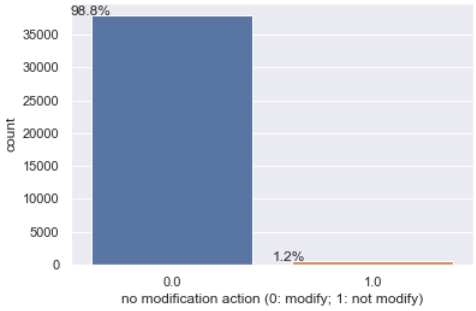


1. "No modification" action analysis

   (first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

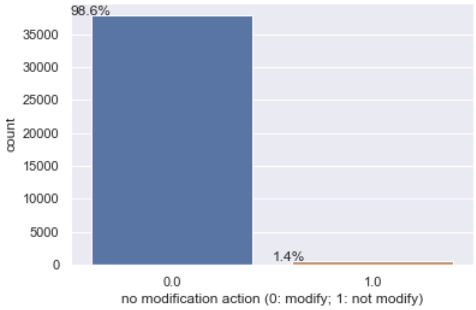


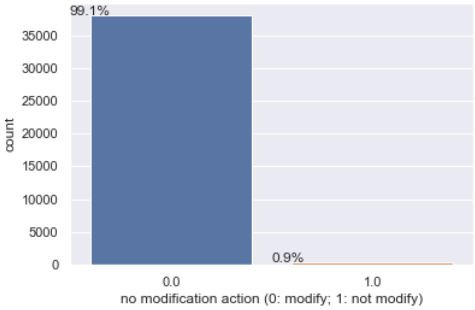




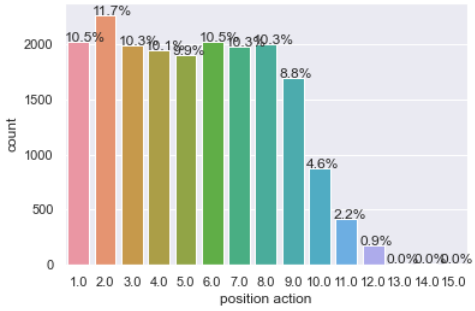
(10 iterations at 3/4)

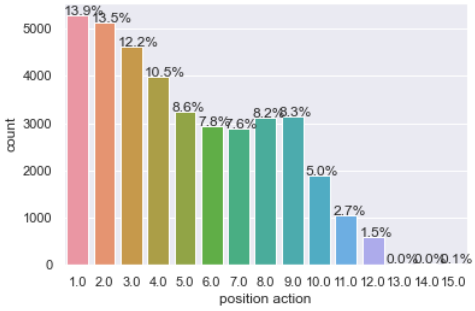
(last 10 iterations)

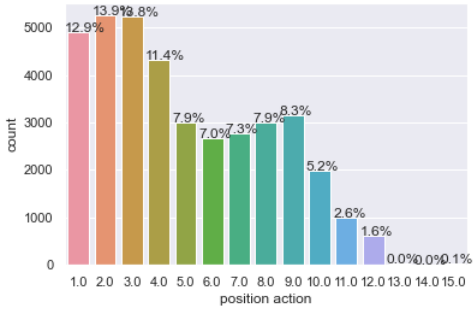


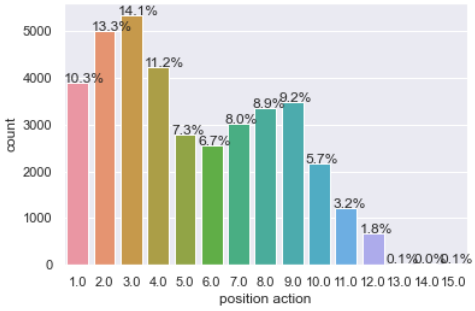


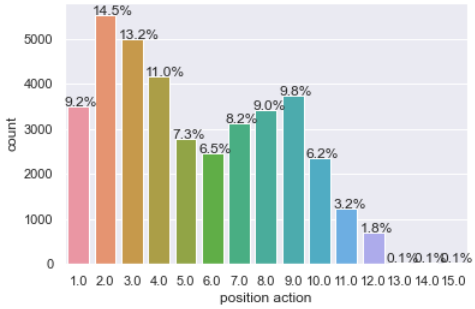
2. Position action analysis



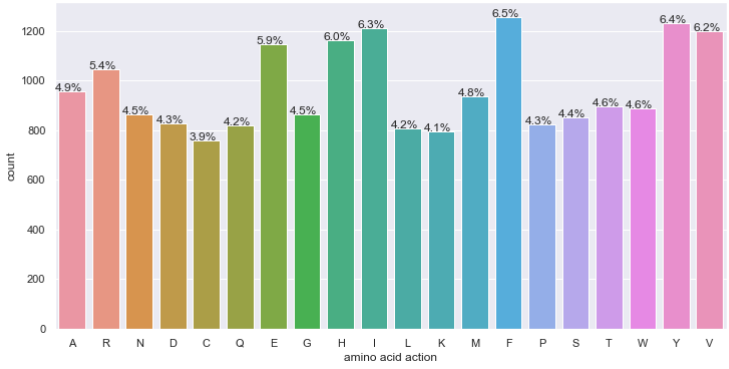


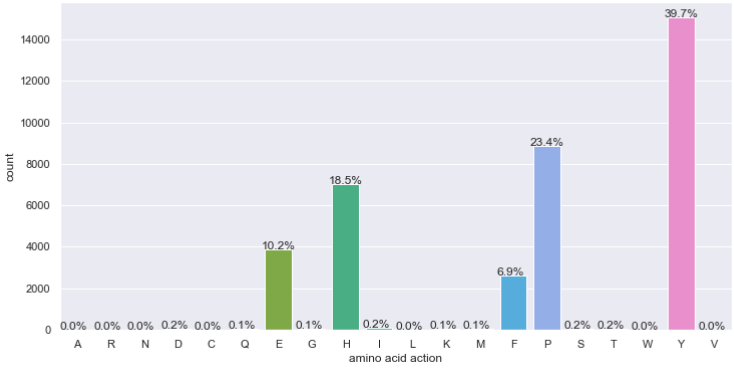


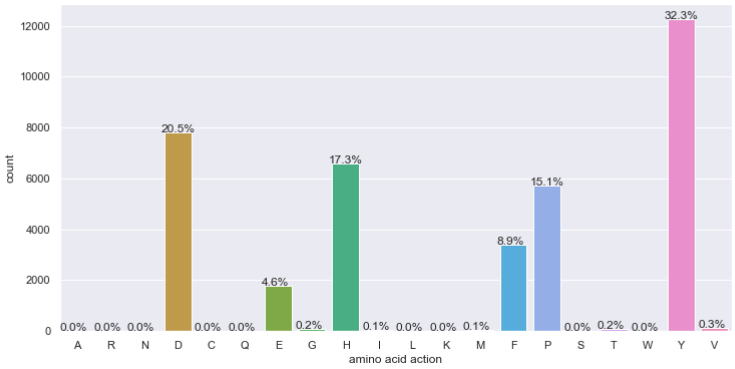


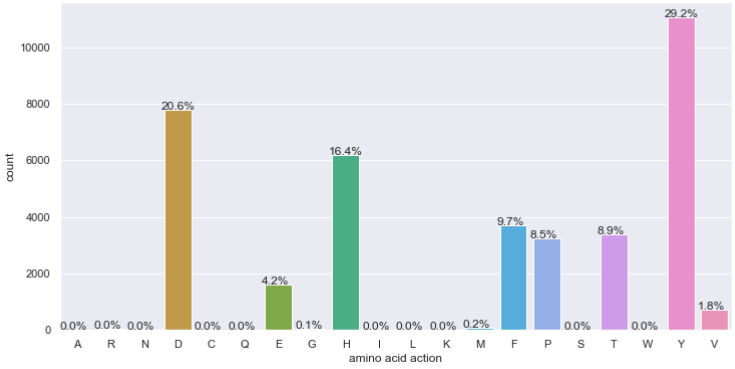


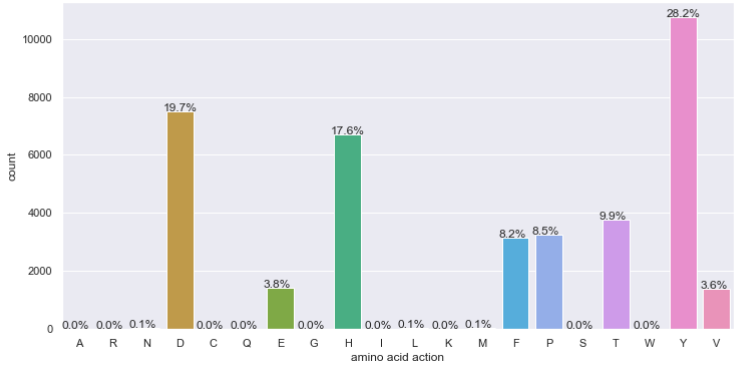
3. Amino action analysis



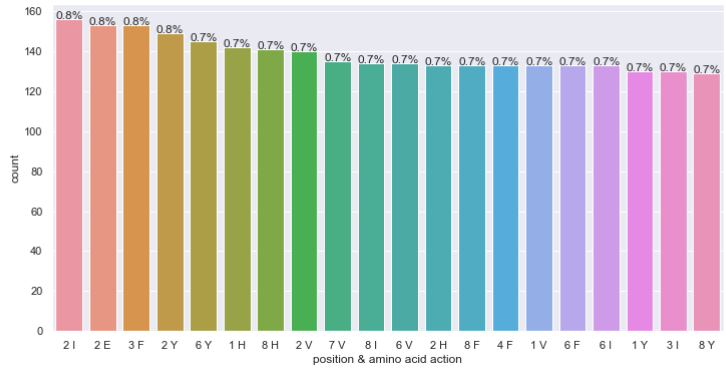


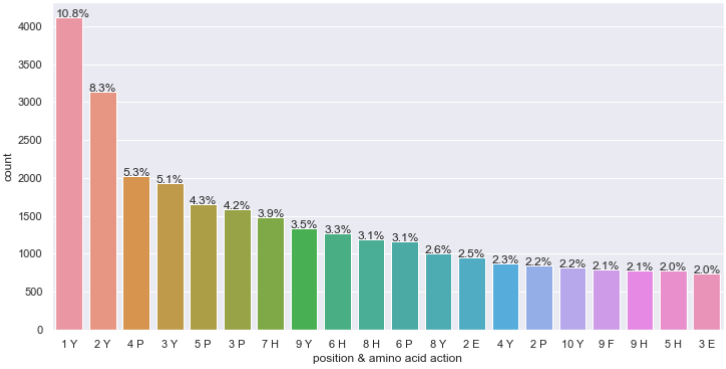


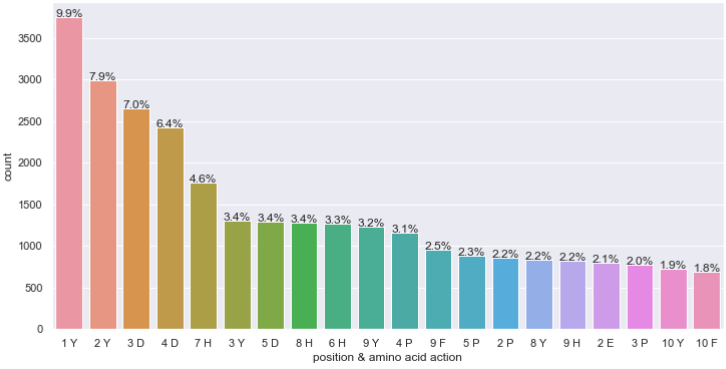


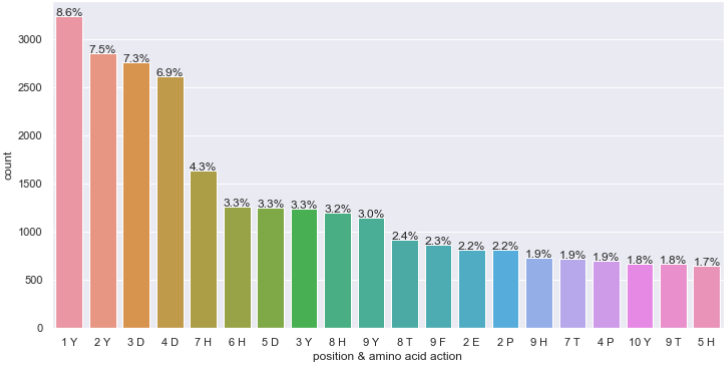


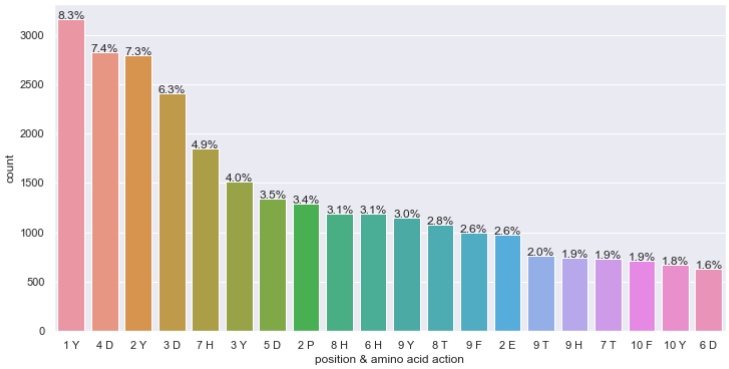
4. Position & Amino action analysis


count
position & amino acid action
0.8% 0.8% 0.8% 0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7%
2 I 2 E 3 F 2 Y 6 Y 1 H 8 H 2 V 7 V 8 I 6 V 2 H 8 F 4 F 1 V 6 F 6 I 1 Y 3 I 8 Y

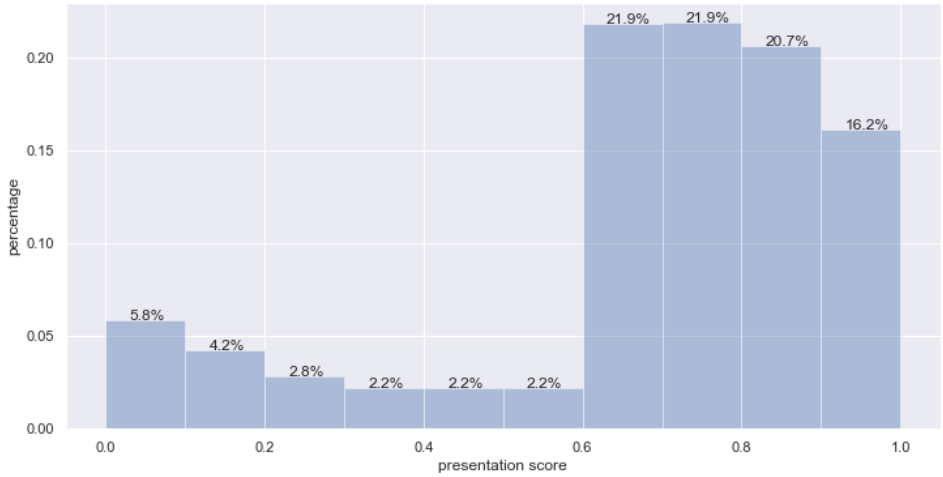

count
position & amino acid action
10.8% 8.3% 5.3% 5.1% 4.3% 4.2% 3.9% 3.5% 3.3% 3.1% 3.1% 2.6% 2.5% 2.3% 2.2% 2.2% 2.1% 2.1% 2.0% 2.0%
1 Y 2 Y 4 P 3 Y 5 P 3 P 7 H 9 Y 6 H 8 H 6 P 8 Y 2 E 4 Y 2 P 10 Y 9 F 9 H 5 H 3 E


count
position & amino acid action
9.9% 7.9% 7.0% 6.4% 4.6% 3.4% 3.4% 3.4% 3.3% 3.2% 3.1% 2.5% 2.3% 2.2% 2.2% 2.2% 2.1% 2.0% 1.9% 1.8%
1 Y 2 Y 3 D 4 D 7 H 3 Y 5 D 8 H 6 H 9 Y 4 P 9 F 5 P 2 P 8 Y 9 H 2 E 3 P 10 Y 10 F

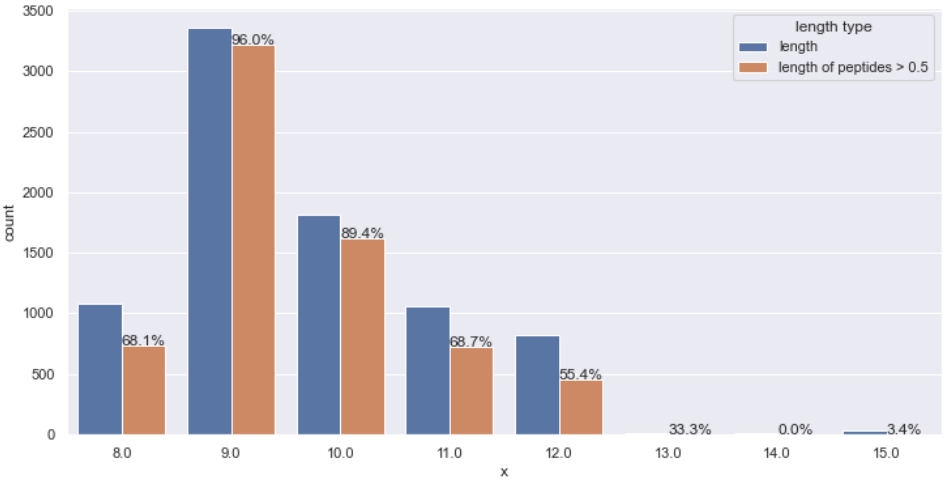

count
position & amino acid action
8.6% 7.5% 7.3% 6.9% 4.3% 3.3% 3.3% 3.3% 3.2% 3.0% 2.4% 2.3% 2.2% 2.2% 1.9% 1.9% 1.9% 1.8% 1.8% 1.7%
1 Y 2 Y 3 D 4 D 7 H 6 H 5 D 3 Y 8 H 9 Y 8 T 9 F 2 E 2 P 9 H 7 T 4 P 10 Y 9 T 5 H



5. Reward analysis

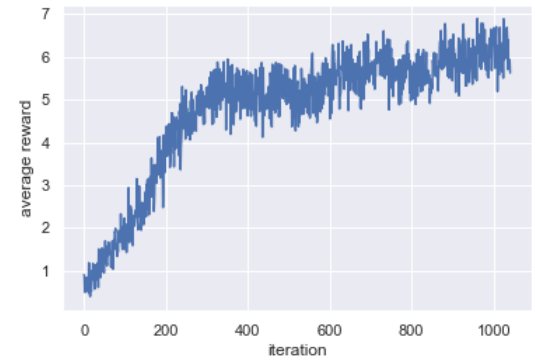


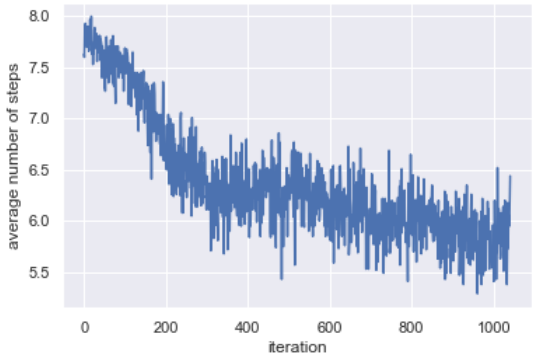
6. Length analysis

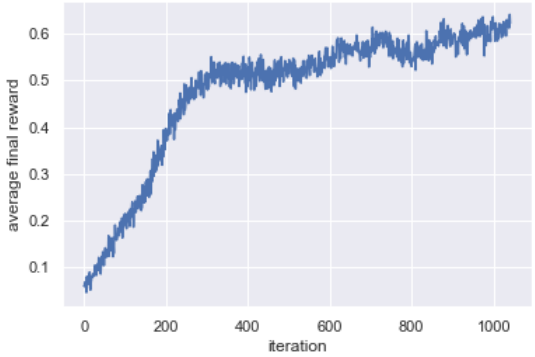


Experiment: stop criteria = 0.75

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

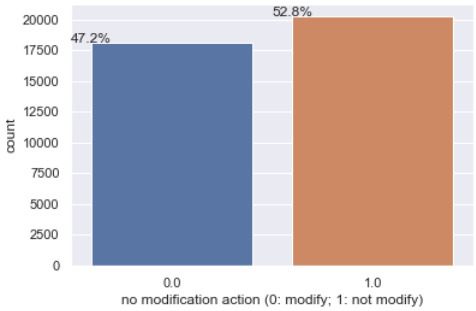


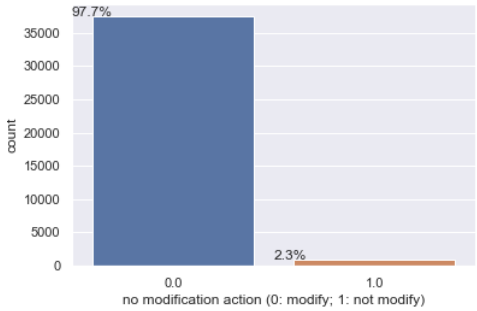


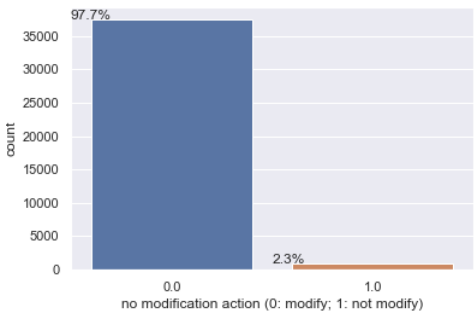


1. "No modification" action analysis

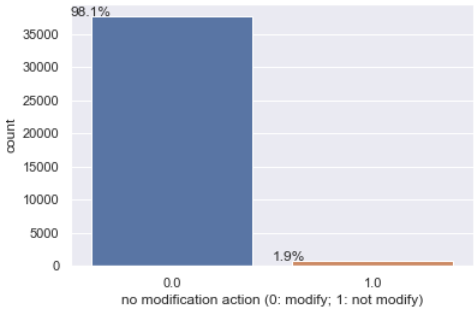
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

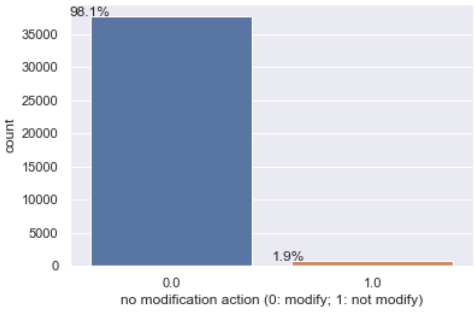




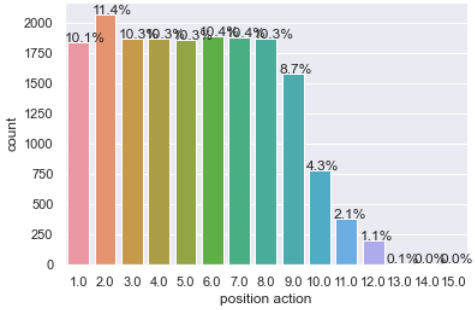


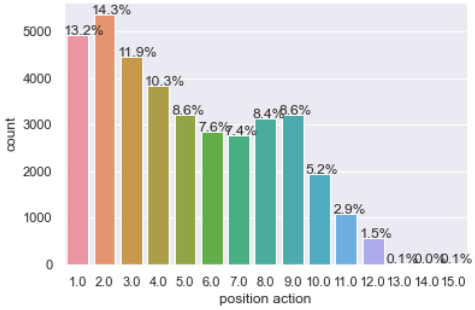
(10 iterations at 3/4) (last 10 iterations)

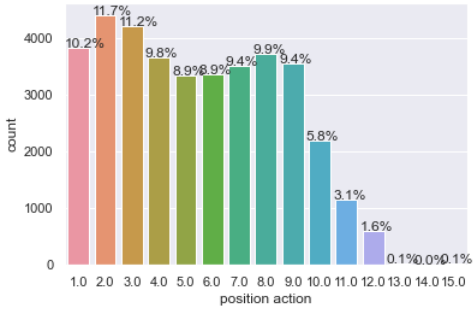


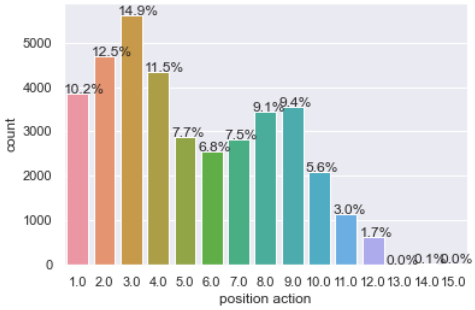


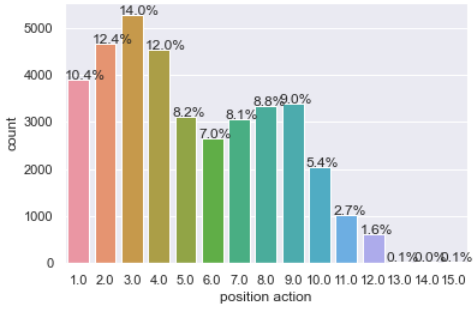
2. Position action analysis



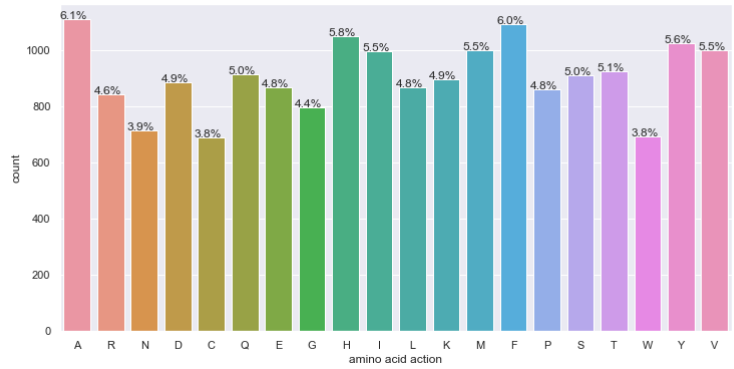


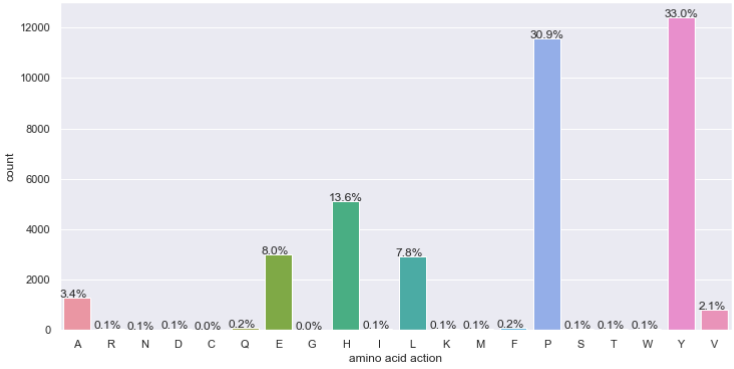


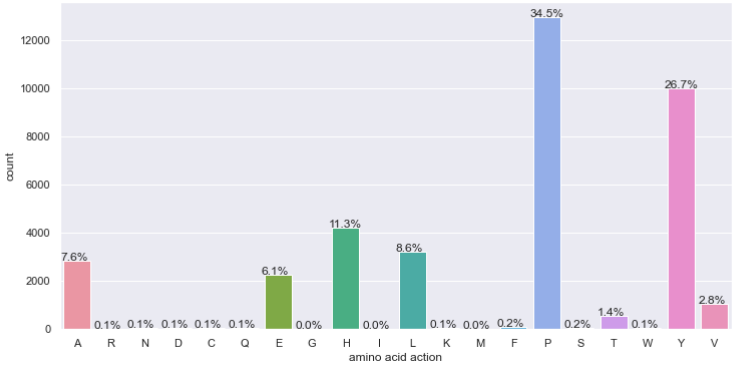


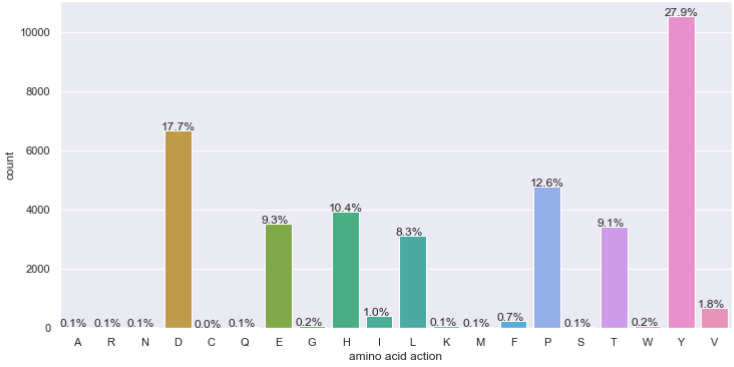


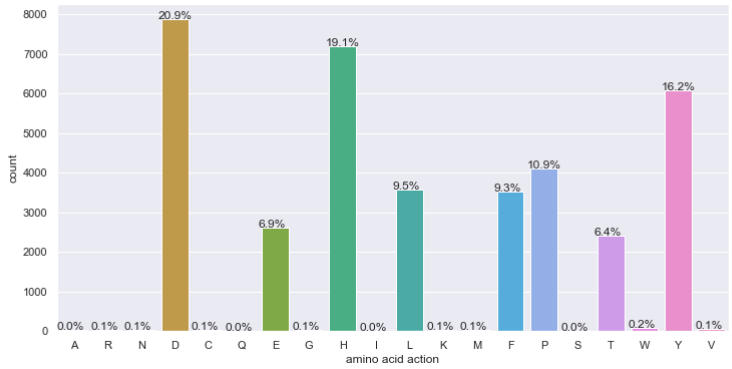
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
6.1% 4.6% 3.9% 4.9% 3.8% 5.0% 4.8% 4.4% 5.8% 5.5% 4.8% 4.9% 5.5% 6.0% 4.8% 5.0% 5.1% 3.8% 5.6% 5.5%

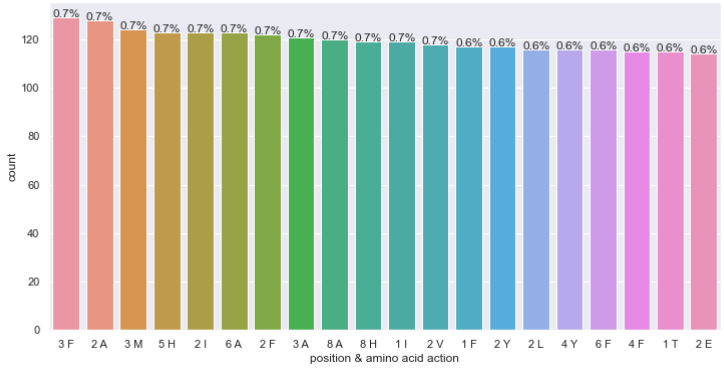

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
3.4% 0.1% 0.1% 0.1% 0.0% 0.2% 8.0% 0.0% 13.6% 0.1% 7.8% 0.1% 0.1% 0.2% 30.9% 0.1% 0.1% 0.1% 33.0% 2.1%

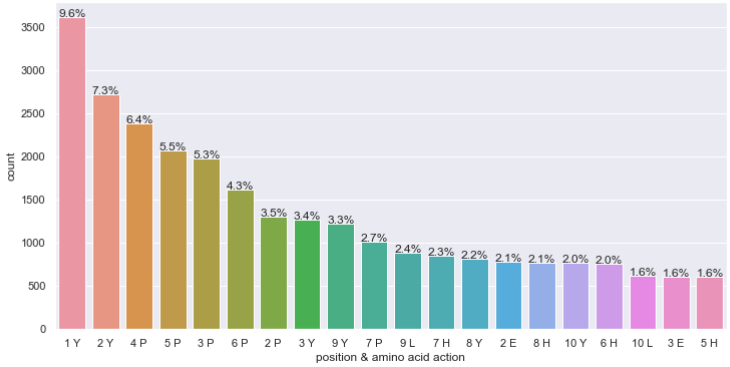

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
7.6% 0.1% 0.1% 0.1% 0.1% 0.1% 6.1% 0.0% 11.3% 0.0% 8.6% 0.1% 0.0% 0.2% 34.5% 0.2% 1.4% 0.1% 26.7% 2.8%

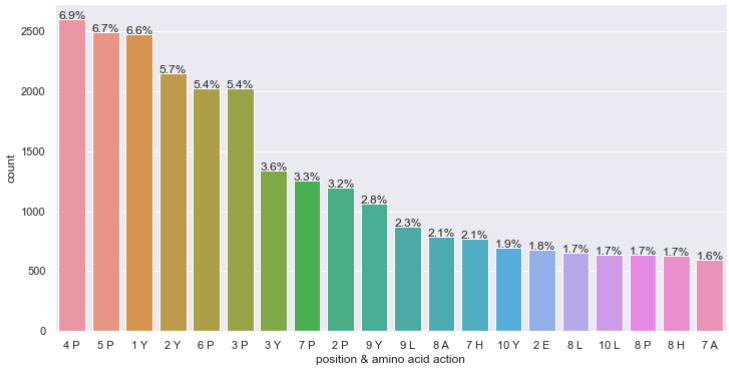

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.1% 0.1% 0.1% 17.7% 0.0% 0.1% 9.3% 0.2% 10.4% 1.0% 8.3% 0.1% 0.1% 0.7% 12.6% 0.1% 9.1% 0.2% 27.9% 1.8%

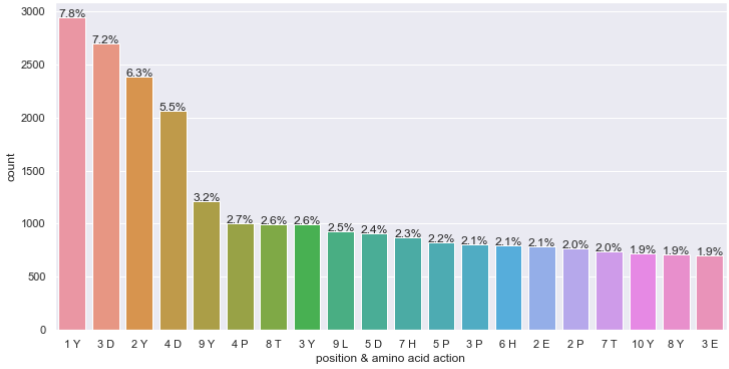


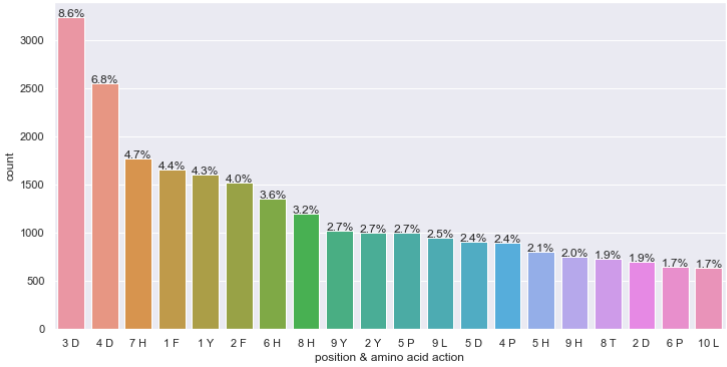
4. Position & Amino action analysis



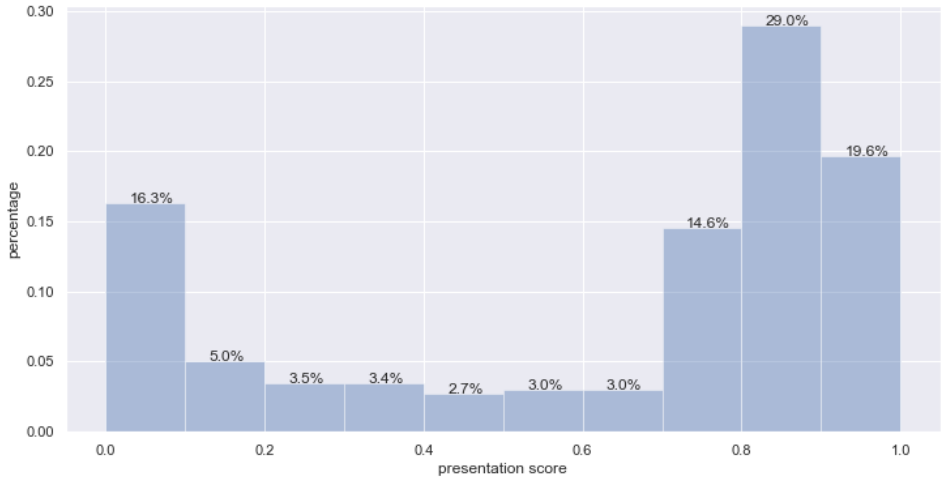




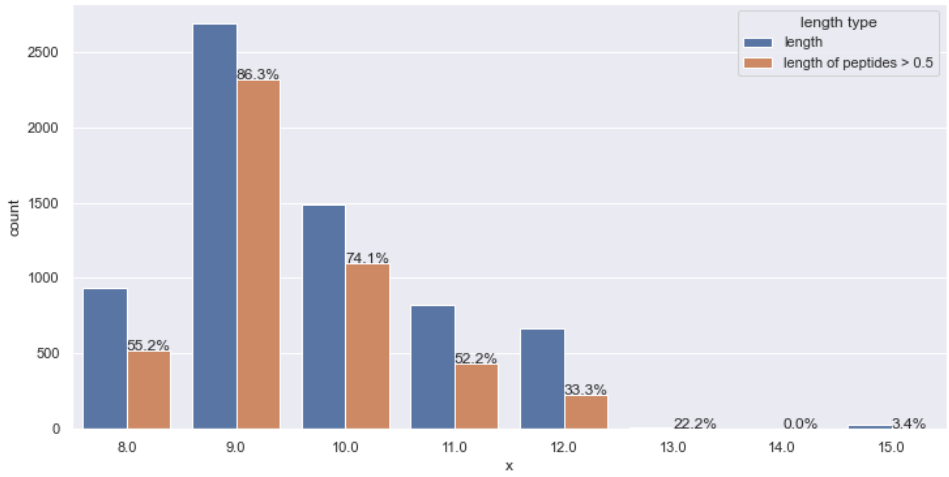




5. Reward analysis

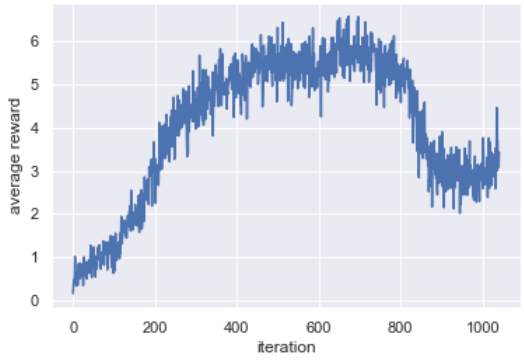


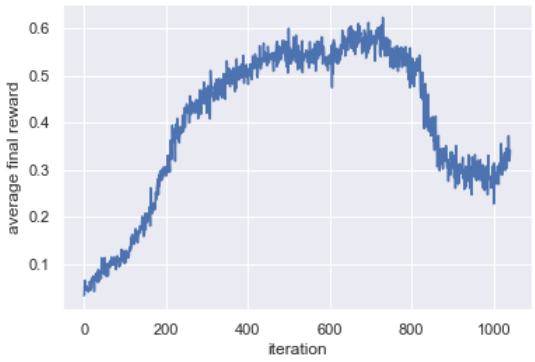
6. Length analysis



Experiment: stop criteria = 1.0

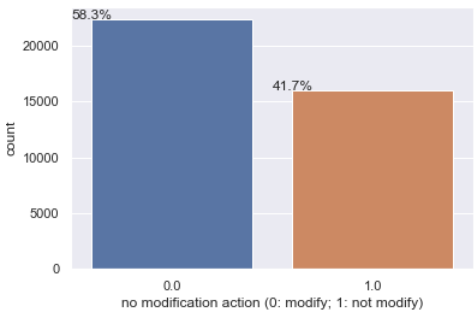
Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

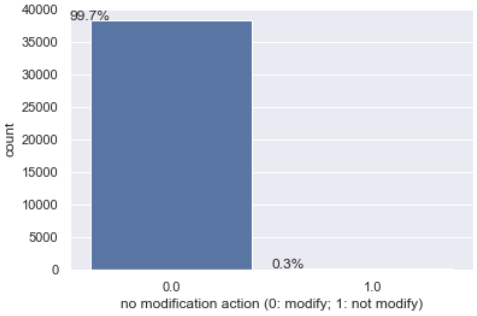


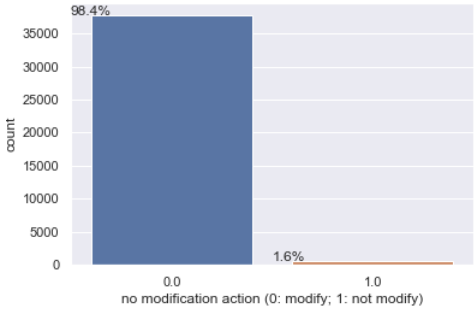


1. "No modification" action analysis

(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

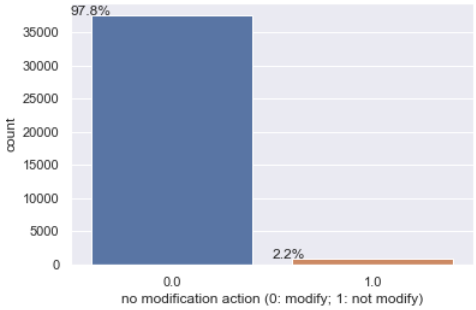


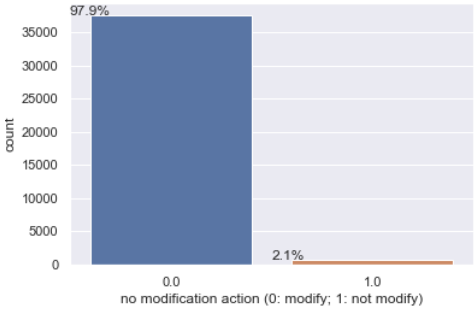




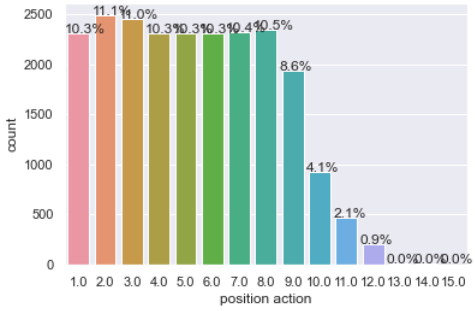
(10 iterations at 3/4)

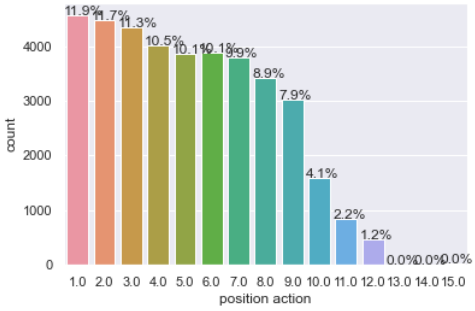
(last 10 iterations)

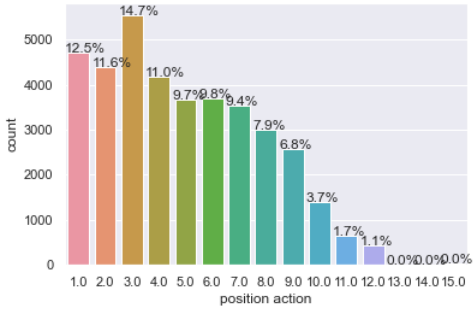


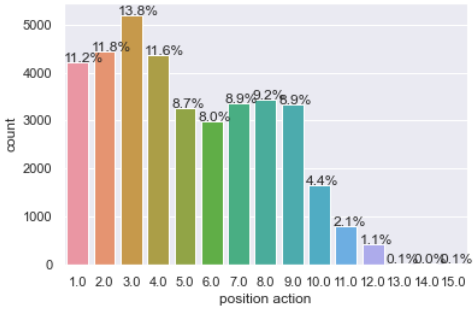


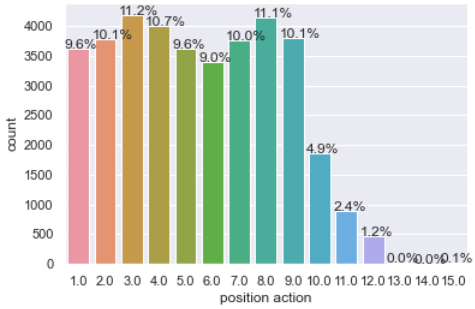
2. Position action analysis



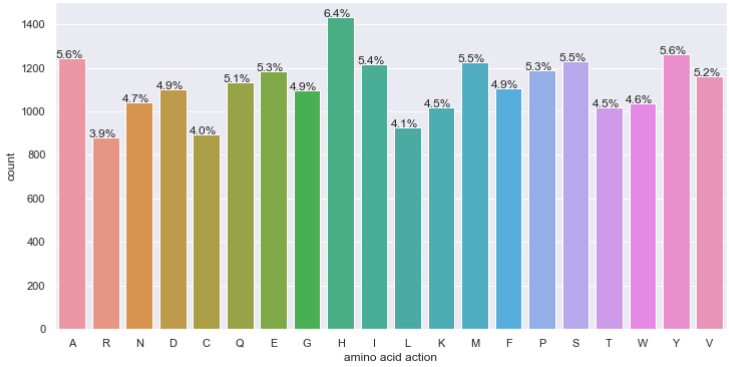


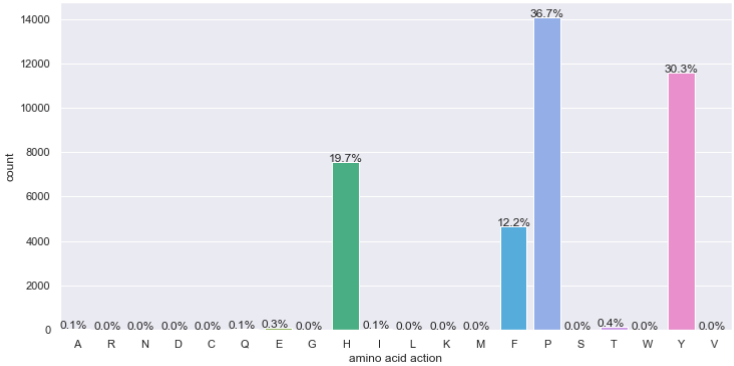


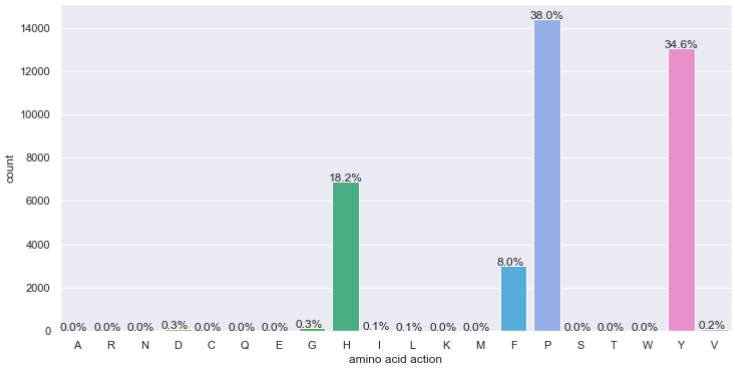


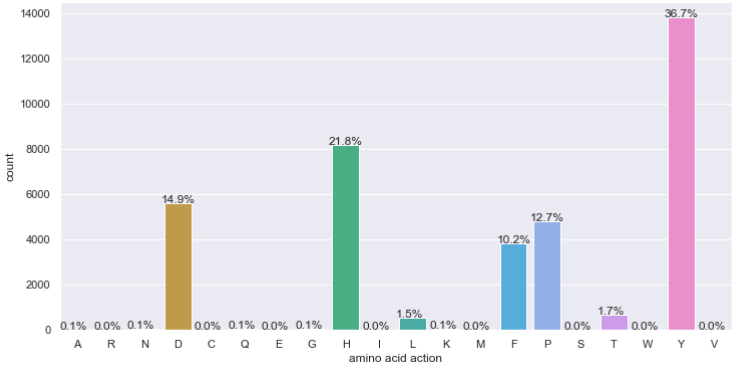


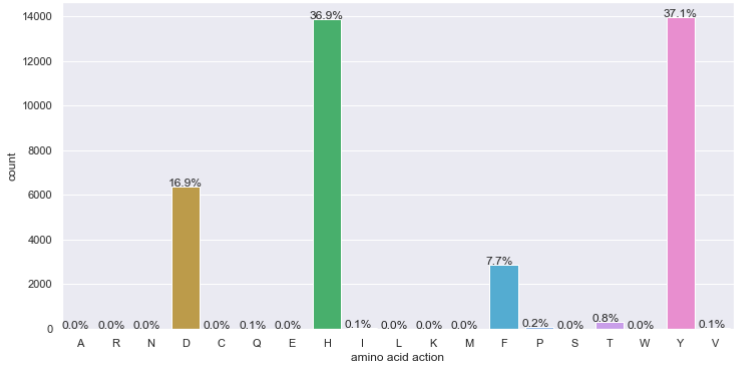
3. Amino action analysis



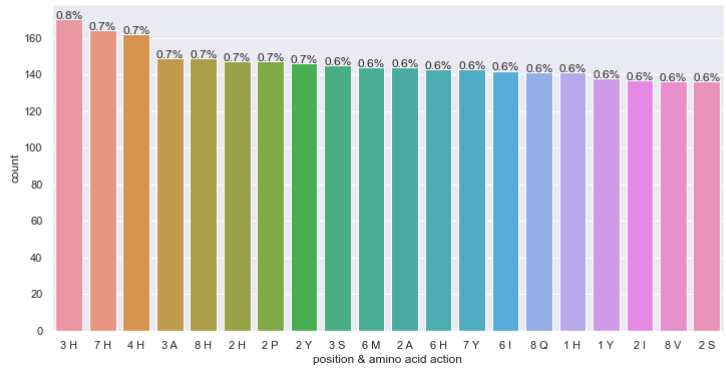


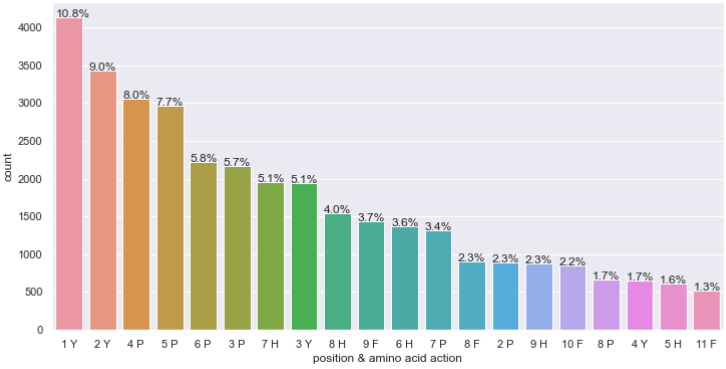


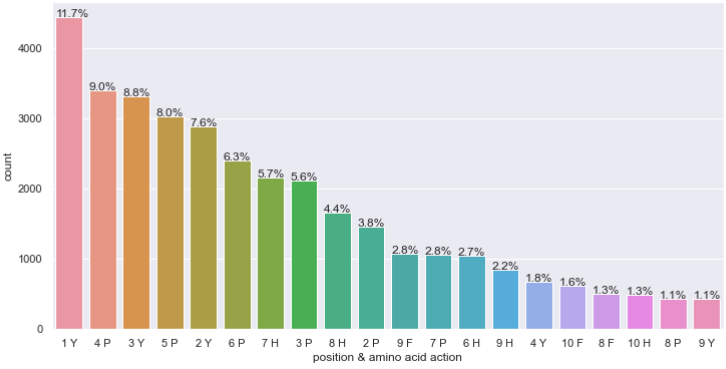


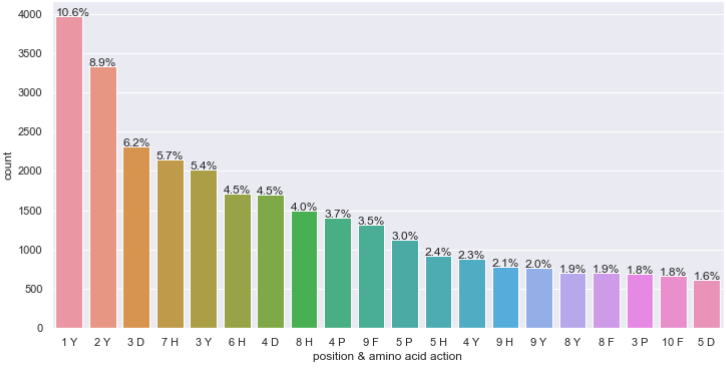


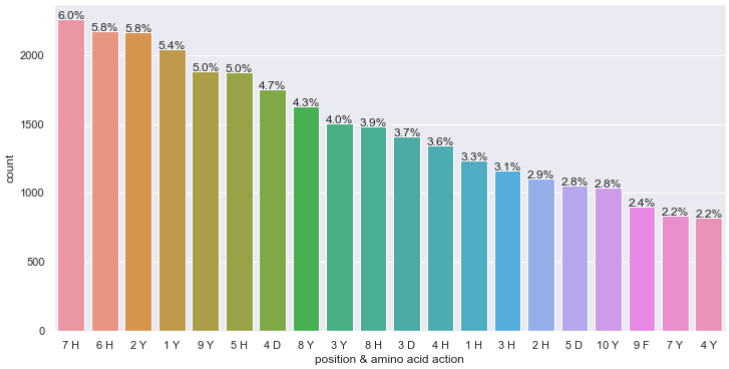
4. Position & Amino action analysis


count
position & amino acid action
0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6% 0.6%
3 H 7 H 4 H 3 A 8 H 2 H 2 P 2 Y 3 S 6 M 2 A 6 H 7 Y 6 I 8 Q 1 H 1 Y 2 I 8 V 2 S

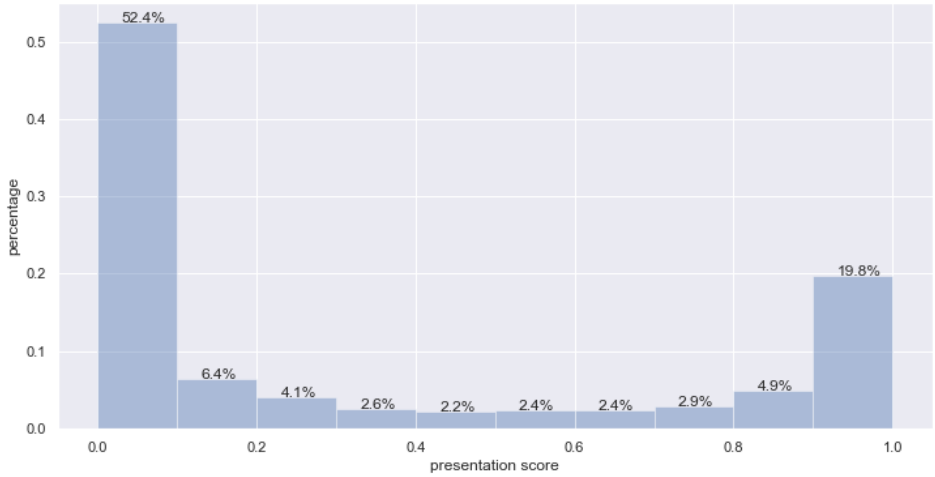

count
position & amino acid action
10.8% 9.0% 8.0% 7.7% 5.8% 5.7% 5.1% 5.1% 4.0% 3.7% 3.6% 3.4% 2.3% 2.3% 2.3% 2.2% 1.7% 1.7% 1.6% 1.3%
1 Y 2 Y 4 P 5 P 6 P 3 P 7 H 3 Y 8 H 9 F 6 H 7 P 8 F 2 P 9 H 10 F 8 P 4 Y 5 H 11 F


count
position & amino acid action
11.7% 9.0% 8.8% 8.0% 7.6% 6.3% 5.7% 5.6% 4.4% 3.8% 2.8% 2.8% 2.7% 2.2% 1.8% 1.6% 1.3% 1.3% 1.1% 1.1%
1 Y 4 P 3 Y 5 P 2 Y 6 P 7 H 3 P 8 H 2 P 9 F 7 P 6 H 9 H 4 Y 10 F 8 F 10 H 8 P 9 Y

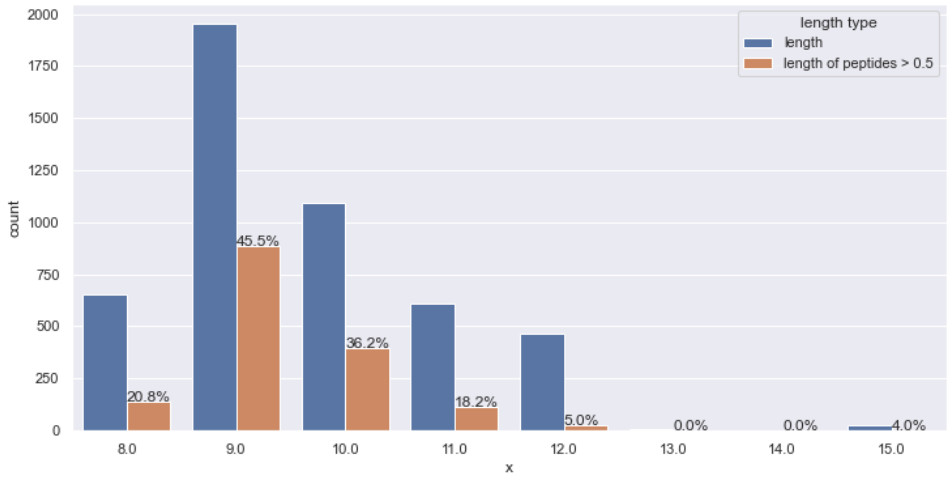

count
position & amino acid action
10.6% 8.9% 6.2% 5.7% 5.4% 4.5% 4.5% 4.0% 3.7% 3.5% 3.0% 2.4% 2.3% 2.1% 2.0% 1.9% 1.9% 1.8% 1.8% 1.6%
1 Y 2 Y 3 D 7 H 3 Y 6 H 4 D 8 H 4 P 9 F 5 P 5 H 4 Y 9 H 9 Y 8 Y 8 F 3 P 10 F 5 D



5. Reward analysis



6. Length analysis



Update (6-2)

Next step:

1) add the constraint that the generated peptides are dissimilar with human peptides. Use as a component of the reward function or as a hard constraint or as a **post-processing** step.

2) write the paper

3) remove "no modification" action

Future plan:

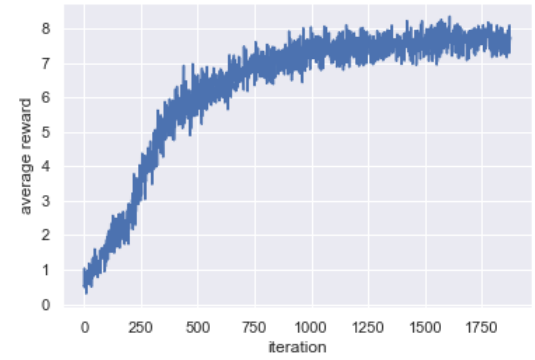
(1) Physics Guided DRL for TCR-pMHC Modeling in the Small Positive Sample Setting

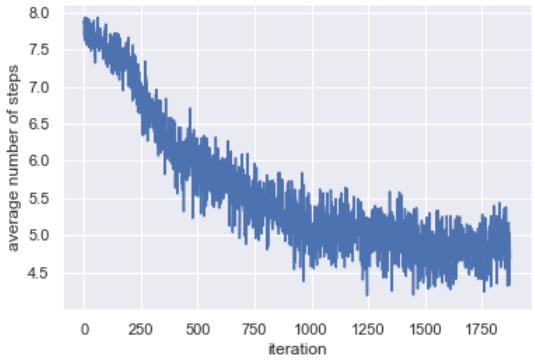
(2) Combine generative classifier with DRL to optimize peptide

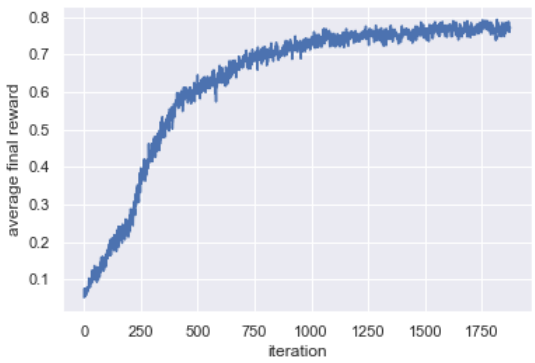
1. Result Analysis with only final rewards

Experiment: stop criteria = 0.75

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

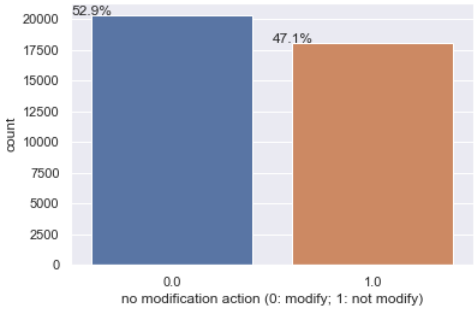


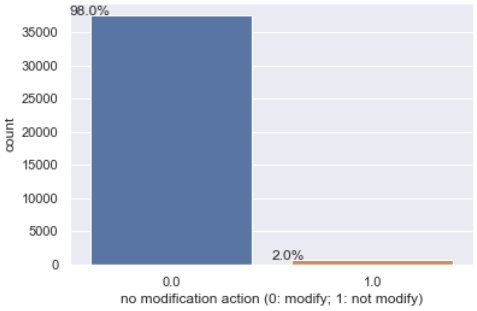


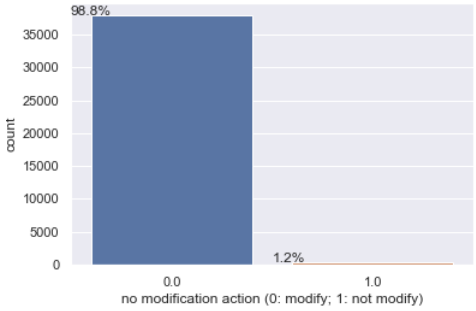


1. "No modification" action analysis

(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

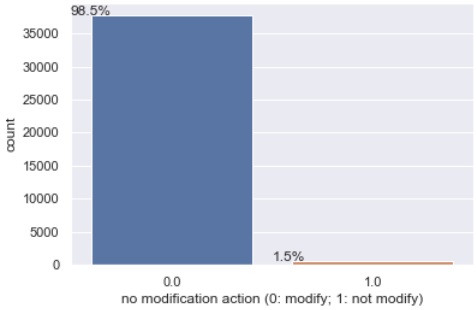


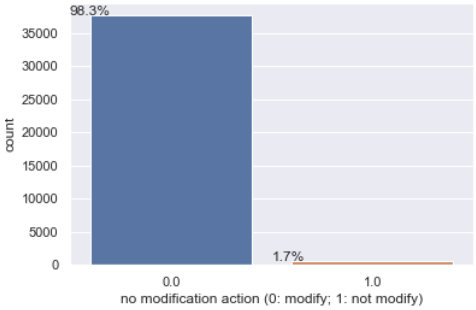




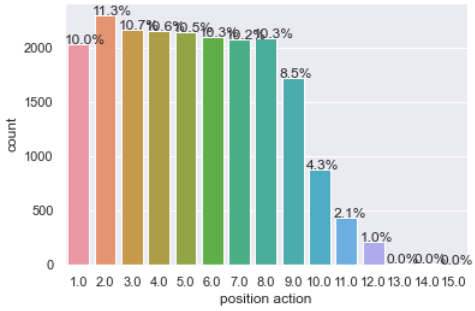
(10 iterations at 3/4)

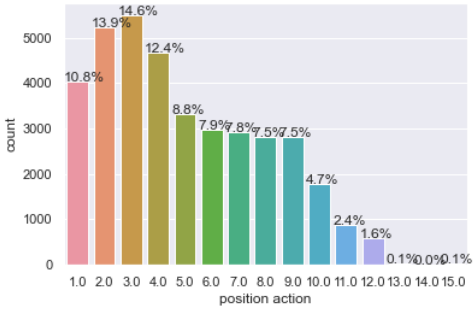
(last 10 iterations)

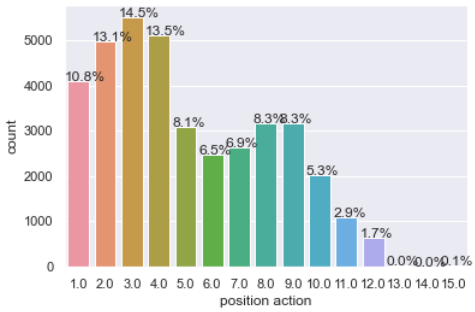


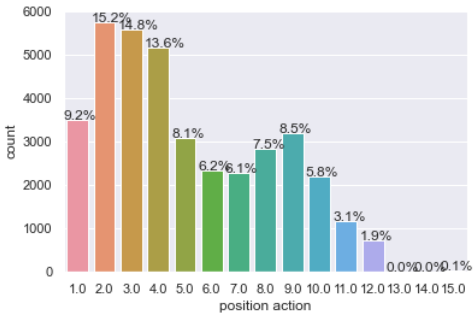


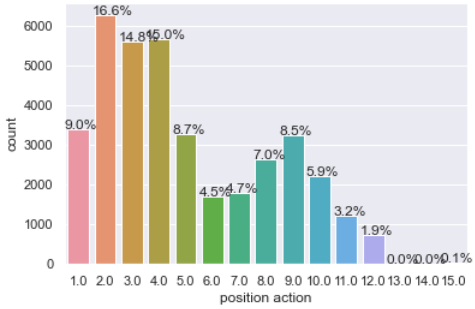
2. Position action analysis



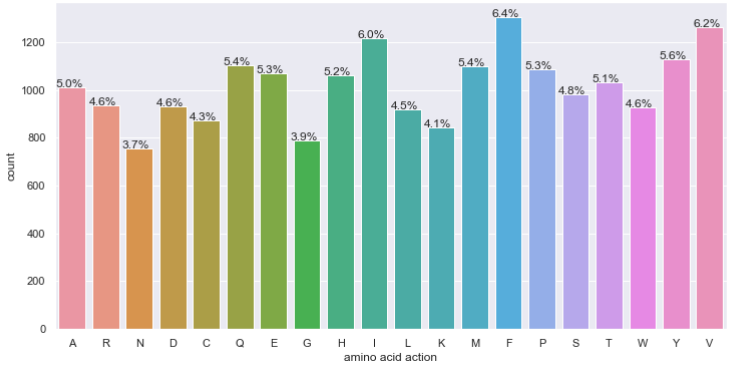


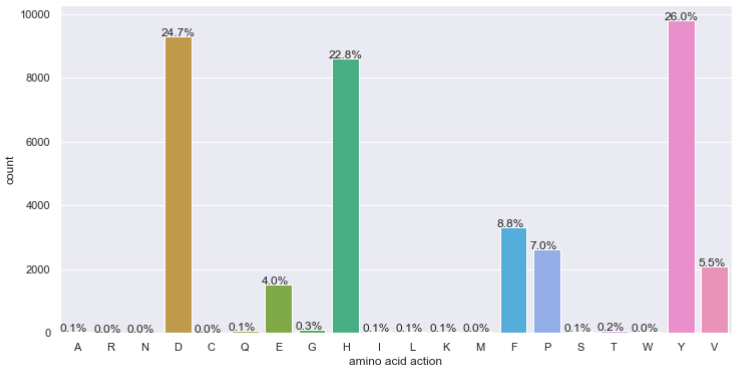


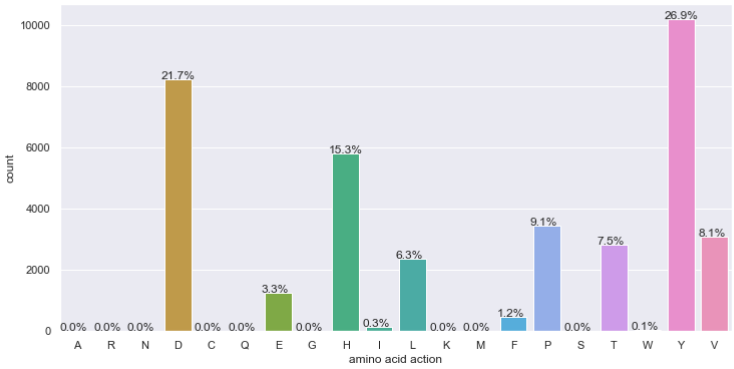


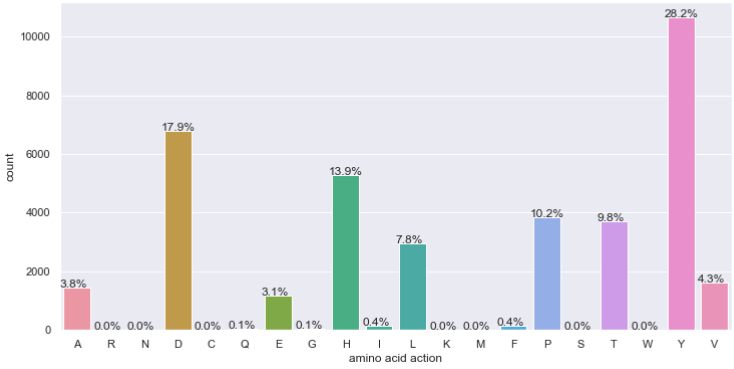


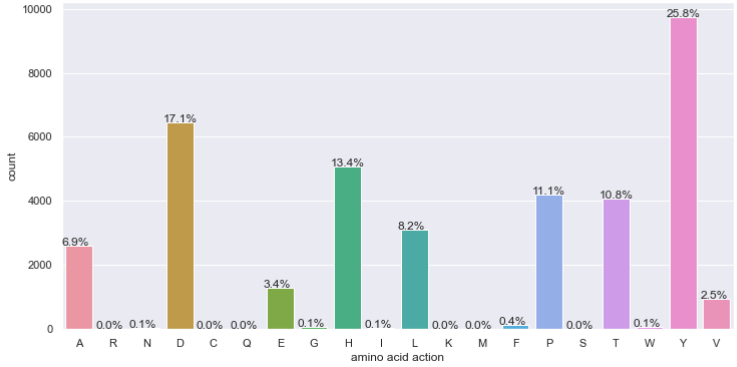
3. Amino action analysis



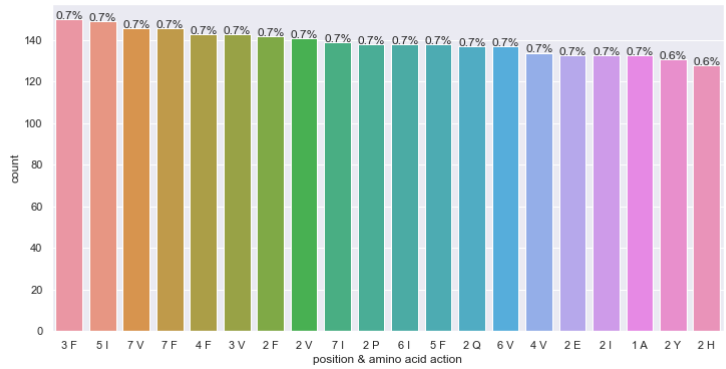


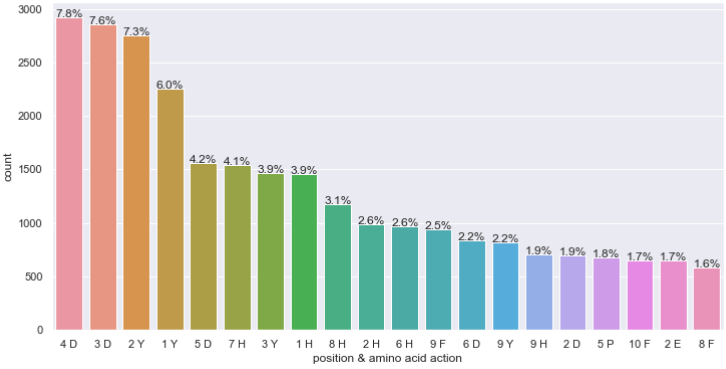


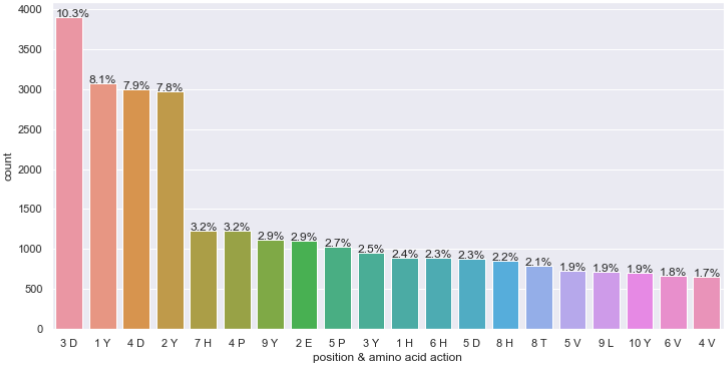


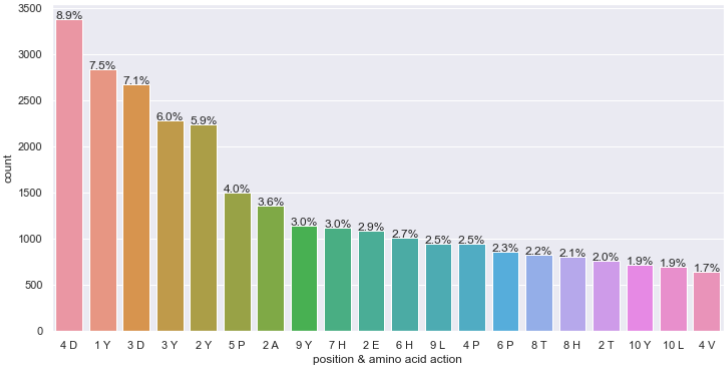


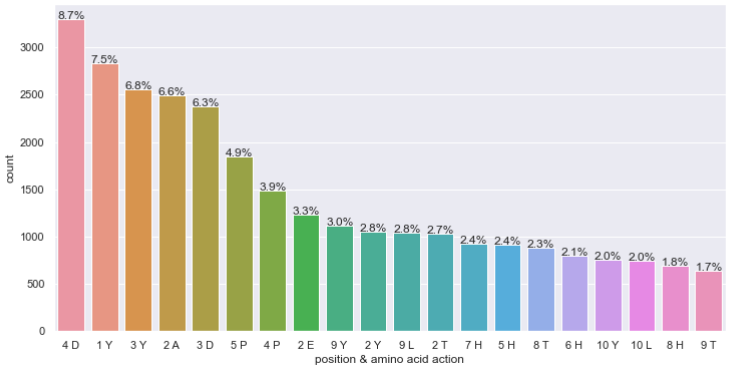
4. Position & Amino action analysis


count
position & amino acid action
3 F 5 I 7 V 7 F 4 F 3 V 2 F 2 V 7 I 2 P 6 I 5 F 2 Q 6 V 4 V 2 E 2 I 1 A 2 Y 2 H
0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.6% 0.6%

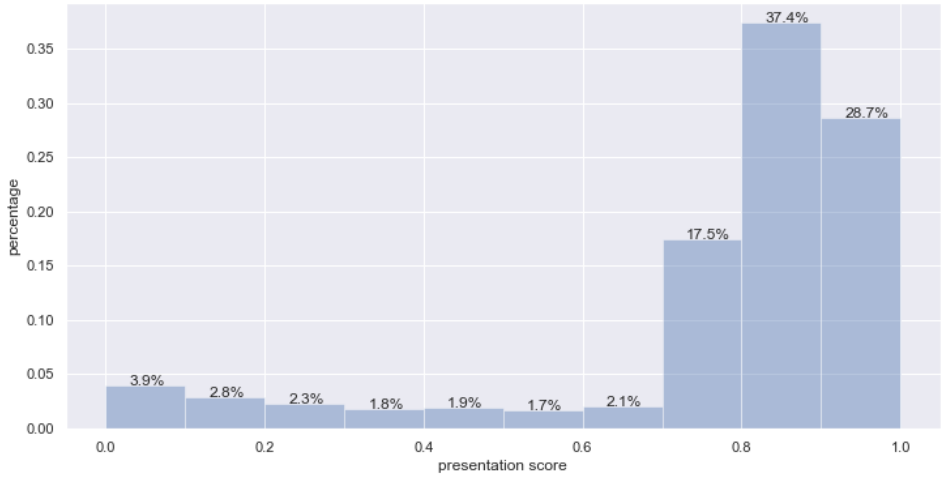

count
position & amino acid action
4 D 3 D 2 Y 1 Y 5 D 7 H 3 Y 1 H 8 H 2 H 6 H 9 F 6 D 9 Y 9 H 2 D 5 P 10 F 2 E 8 F
7.8% 7.6% 7.3% 6.0% 4.2% 4.1% 3.9% 3.9% 3.1% 2.6% 2.6% 2.5% 2.2% 2.2% 1.9% 1.9% 1.8% 1.7% 1.7% 1.6%


count
position & amino acid action
3 D 1 Y 4 D 2 Y 7 H 4 P 9 Y 2 E 5 P 3 Y 1 H 6 H 5 D 8 H 8 T 5 V 9 L 10 Y 6 V 4 V
10.3% 8.1% 7.9% 7.8% 3.2% 3.2% 2.9% 2.9% 2.7% 2.5% 2.4% 2.3% 2.3% 2.2% 2.1% 1.9% 1.9% 1.9% 1.8% 1.7%

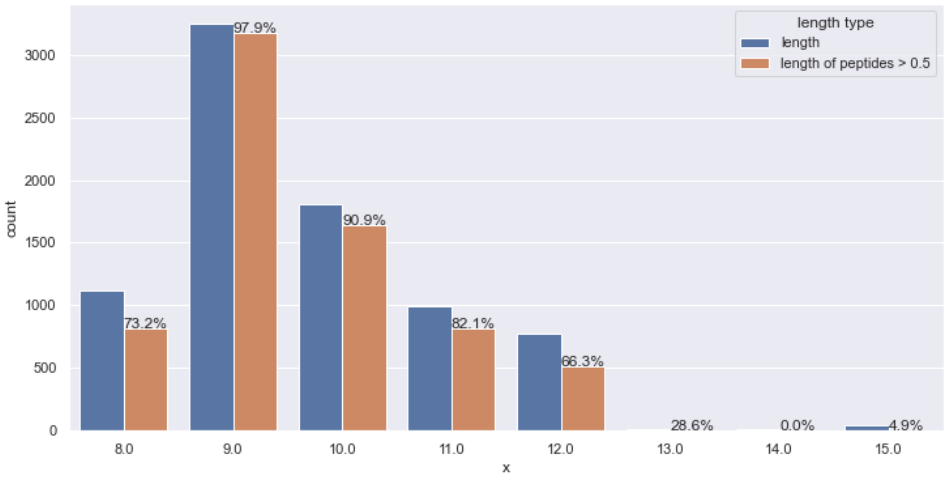

count
position & amino acid action
4 D 1 Y 3 D 3 Y 2 Y 5 P 2 A 9 Y 7 H 2 E 6 H 9 L 4 P 6 P 8 T 8 H 2 T 10 Y 10 L 4 V
8.9% 7.5% 7.1% 6.0% 5.9% 4.0% 3.6% 3.0% 3.0% 2.9% 2.7% 2.5% 2.5% 2.3% 2.2% 2.1% 2.0% 1.9% 1.9% 1.7%



5. Reward analysis

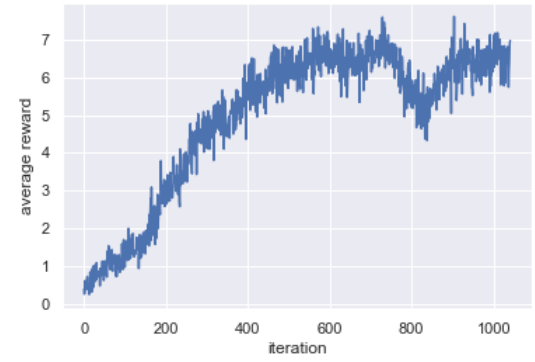


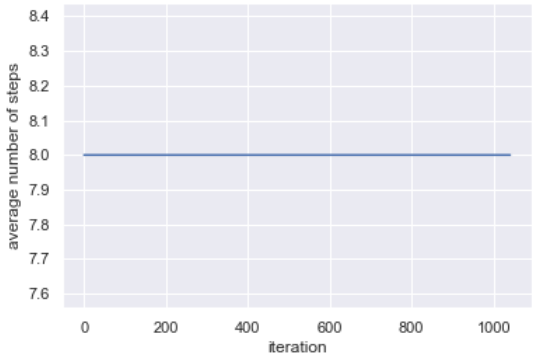
6. Length analysis

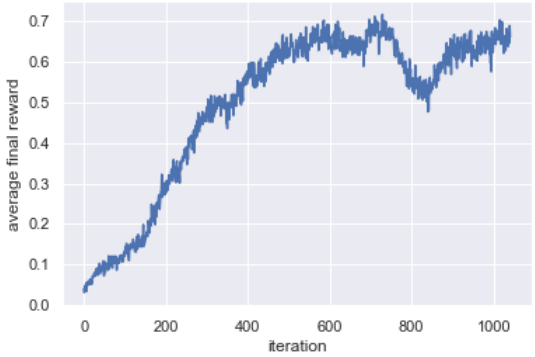


Experiment: stop criteria = 1.0

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

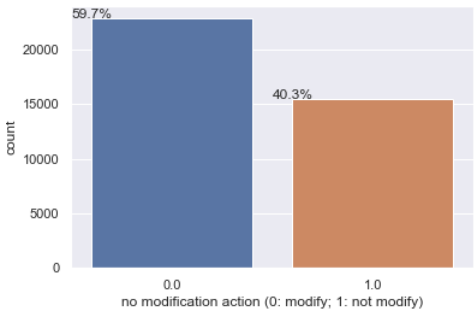


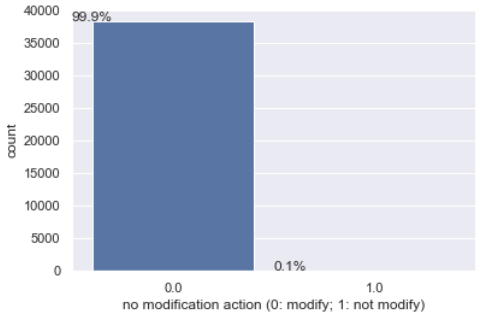


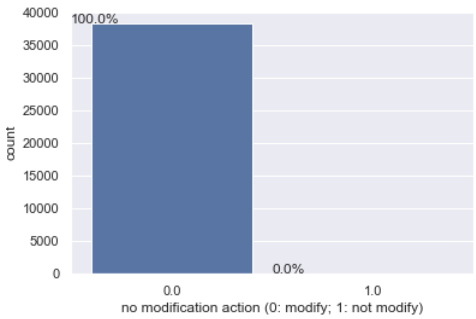


1. "No modification" action analysis

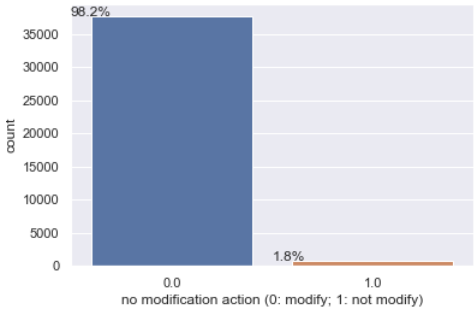
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

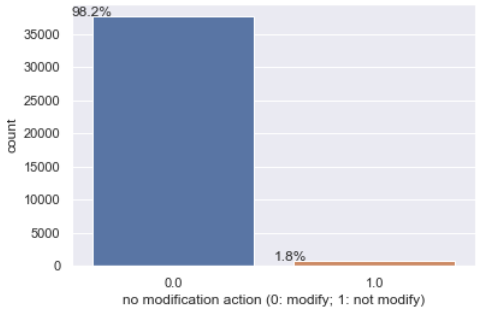




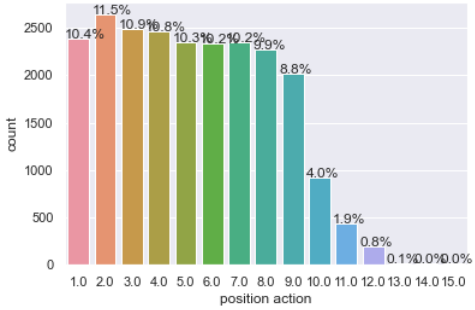


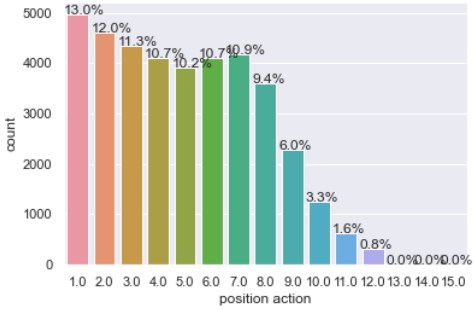
(10 iterations at 3/4) (last 10 iterations)

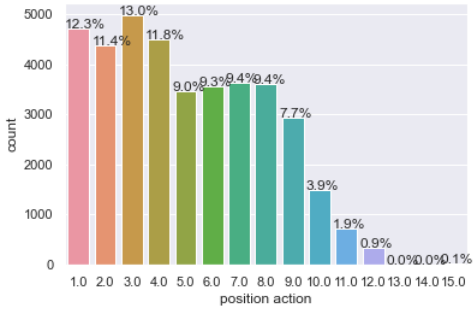


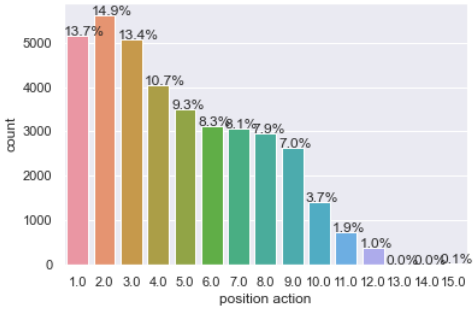


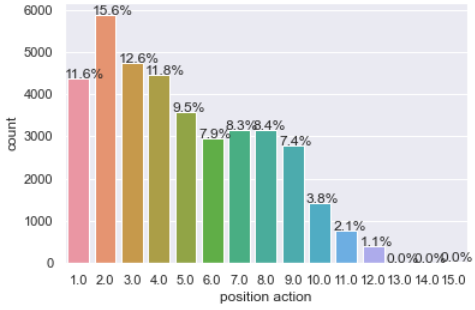
2. Position action analysis



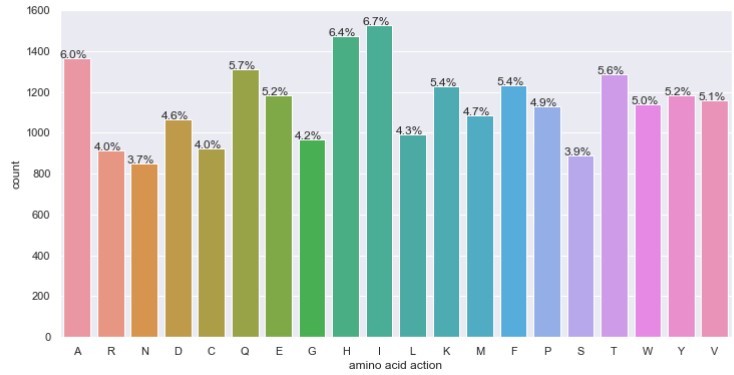


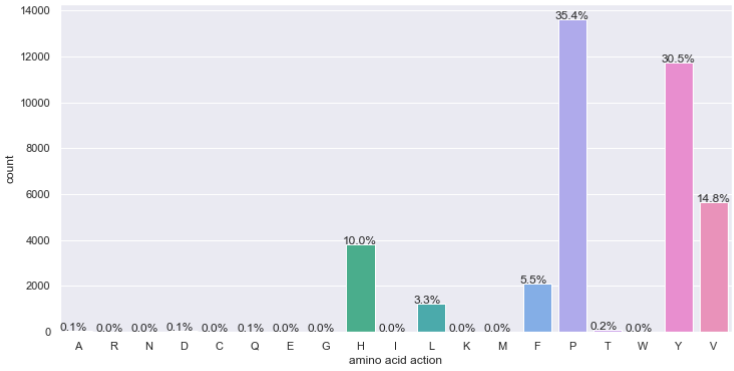


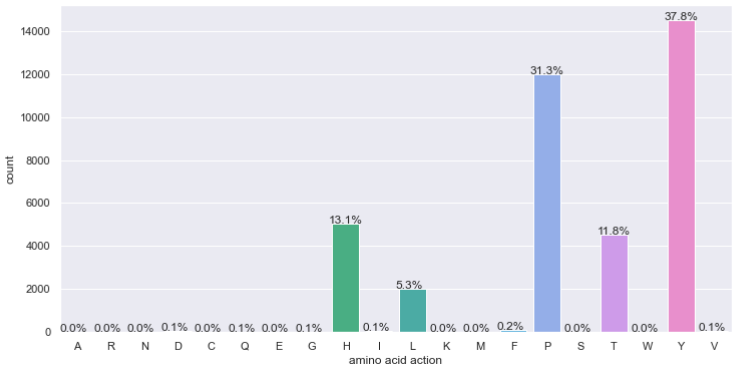


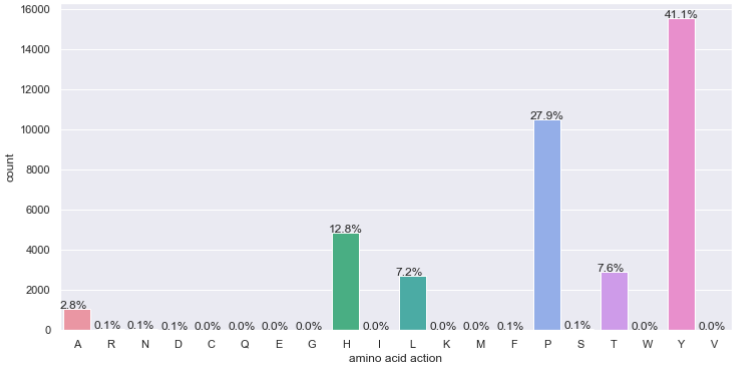


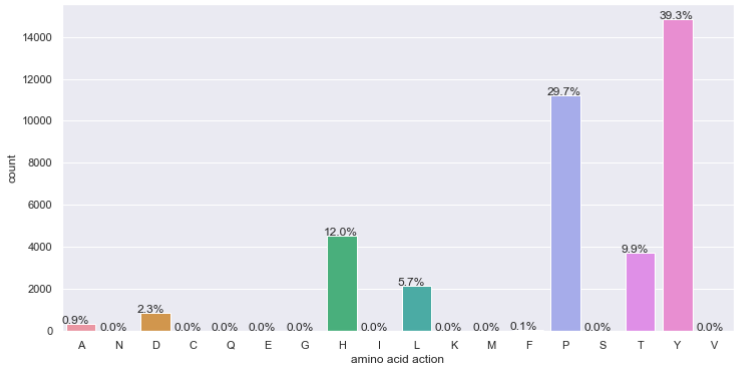
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
6.0% 4.0% 3.7% 4.6% 4.0% 5.7% 5.2% 4.2% 6.4% 6.7% 4.3% 5.4% 4.7% 5.4% 4.9% 3.9% 5.6% 5.0% 5.2% 5.1%

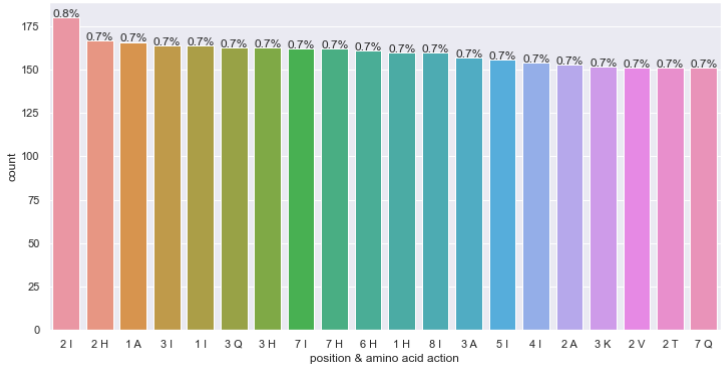

count
amino acid action
A R N D C Q E G H I L K M F P T W Y V
0.1% 0.0% 0.0% 0.1% 0.0% 0.1% 0.0% 0.0% 10.0% 0.0% 3.3% 0.0% 0.0% 5.5% 35.4% 0.2% 0.0% 30.5% 14.8%

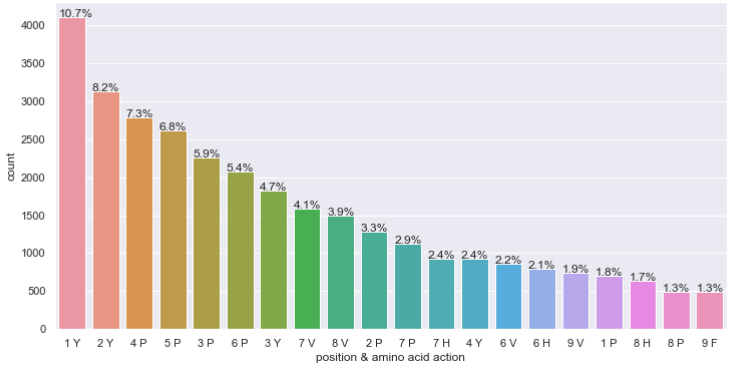

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.0% 0.0% 0.1% 0.0% 0.1% 0.0% 0.1% 13.1% 0.1% 5.3% 0.0% 0.0% 0.2% 31.3% 0.0% 11.8% 0.0% 37.8% 0.1%

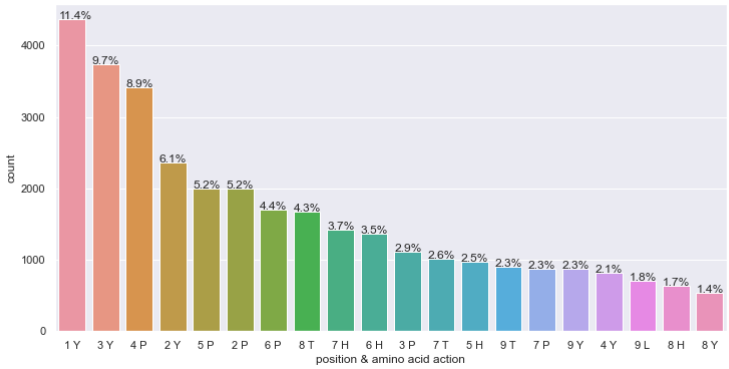

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
2.8% 0.1% 0.1% 0.1% 0.0% 0.0% 0.0% 0.0% 12.8% 0.0% 7.2% 0.0% 0.0% 0.1% 27.9% 0.1% 7.6% 0.0% 41.1% 0.0%

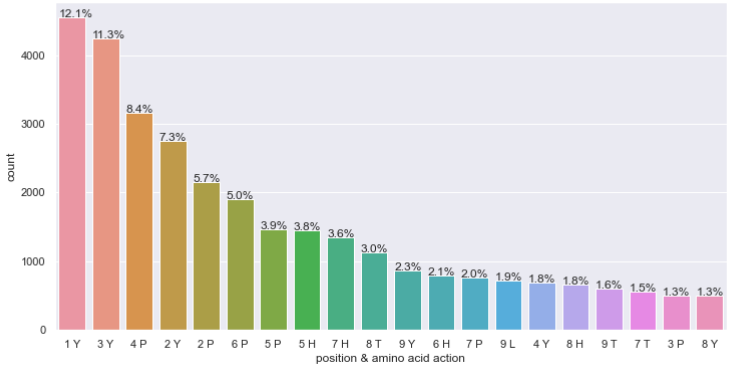


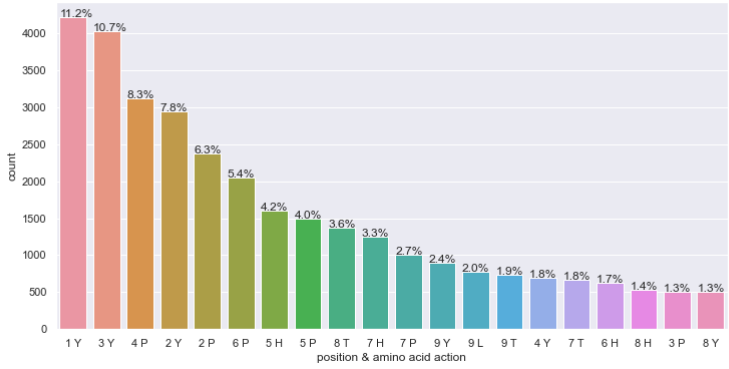
4. Position & Amino action analysis



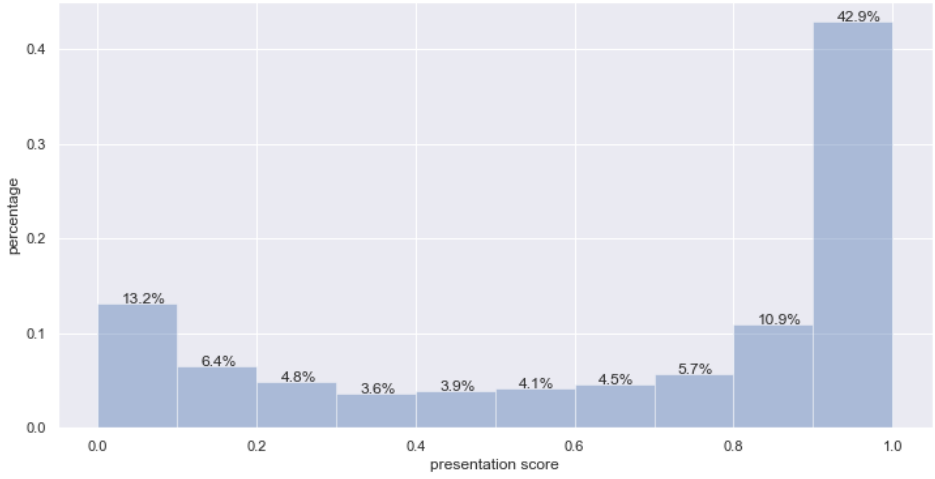




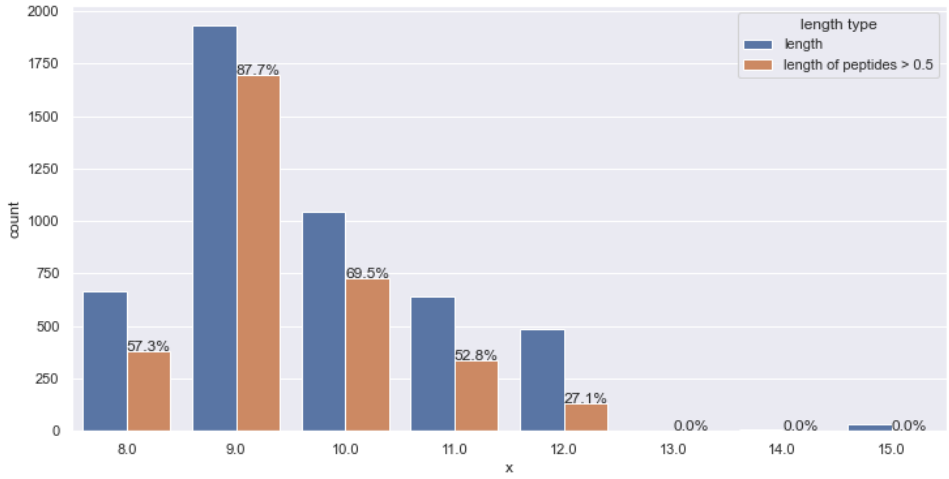




5. Reward analysis



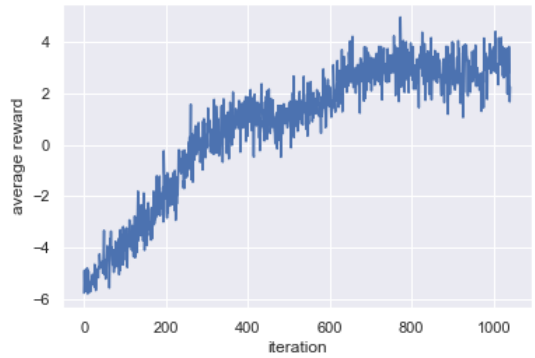
6. Length analysis

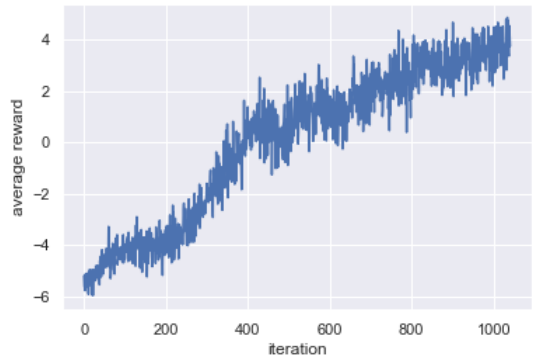


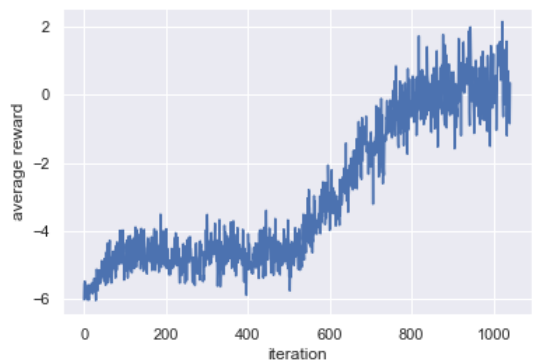
2. Prioritized Experience Replay

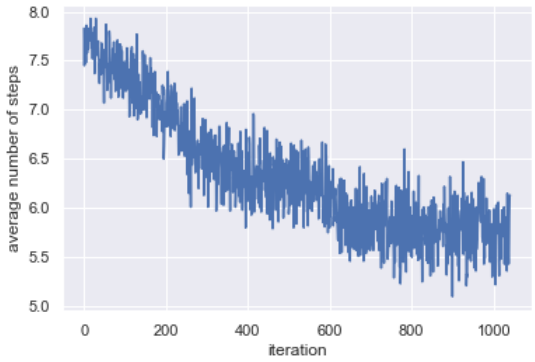
Experiment: immediate reward

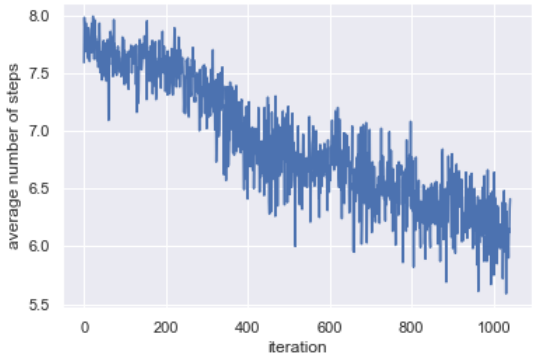
stop criteria = 0.6 stop criteria = 0.75
stop criteria = 1.0

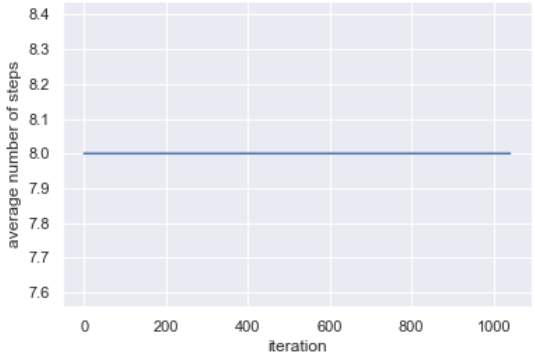


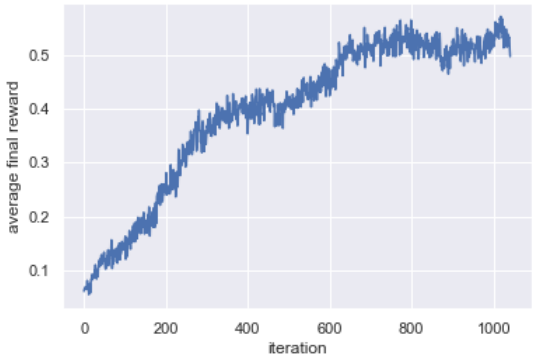


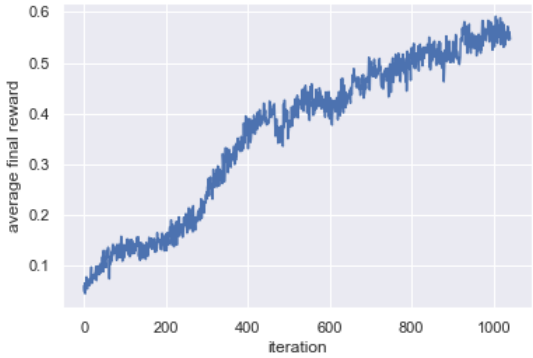


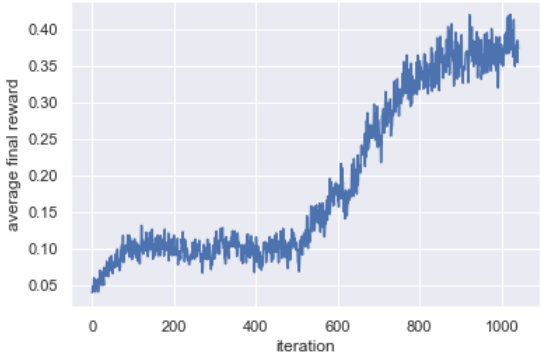

8.0
7.5
7.0
6.5
6.0
5.5
5.0
average number of steps
0
200
400
600
800
1000
iteration


8.0
7.5
7.0
6.5
6.0
5.5
average number of steps
0
200
400
600
800
1000
iteration


8.4
8.3
8.2
8.1
8.0
7.9
7.8
7.7
7.6
average number of steps
0
200
400
600
800
1000
iteration

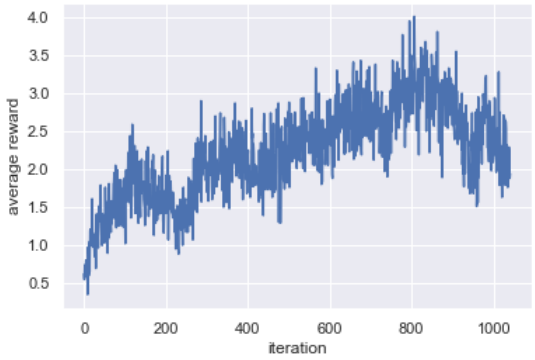


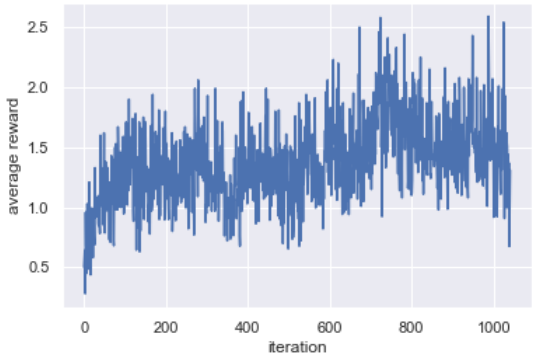


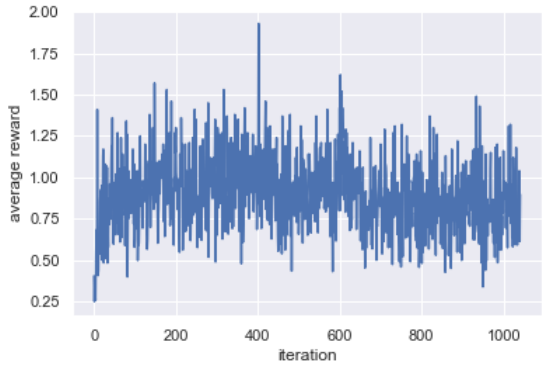


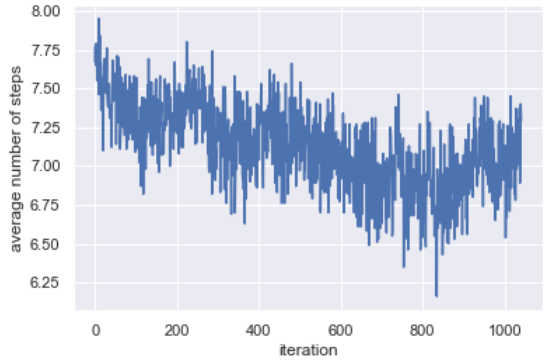
Experiment: only final reward

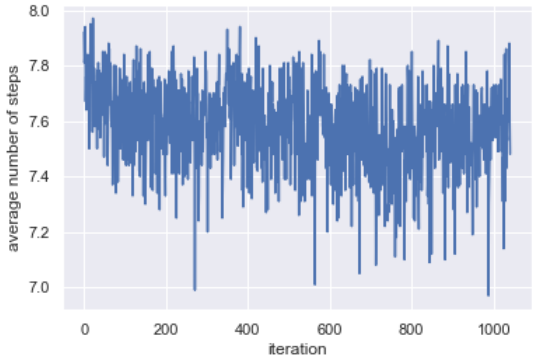
stop criteria = 0.6 stop criteria = 0.75
stop criteria = 1.0

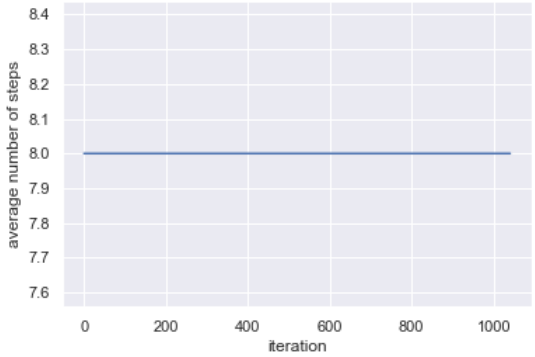

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
average reward
0
200
400
600
800
1000
iteration

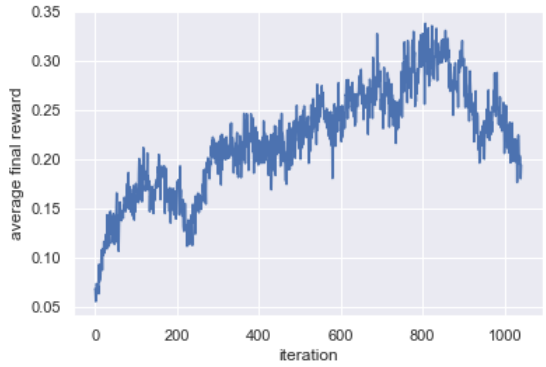

2.5
2.0
1.5
1.0
0.5
average reward
0
200
400
600
800
1000
iteration

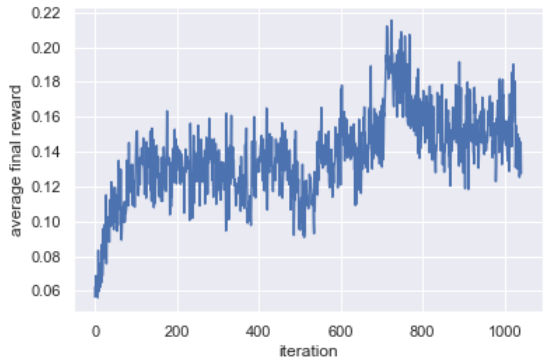

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
average reward
0
200
400
600
800
1000
iteration

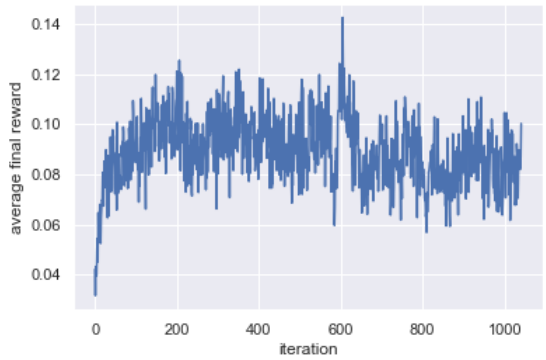

8.00
7.75
7.50
7.25
7.00
6.75
6.50
6.25
average number of steps
0
200
400
600
800
1000
iteration


8.0
7.8
7.6
7.4
7.2
7.0
average number of steps
0
200
400
600
800
1000
iteration


8.4
8.3
8.2
8.1
8.0
7.9
7.8
7.7
7.6
average number of steps
0
200
400
600
800
1000
iteration



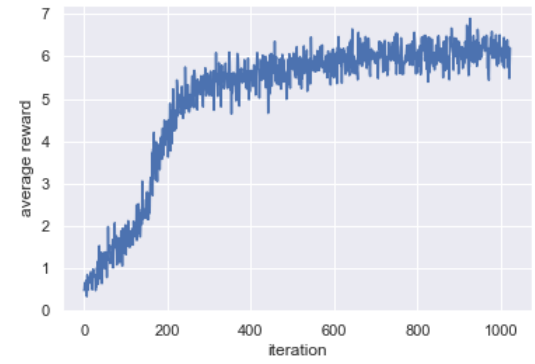


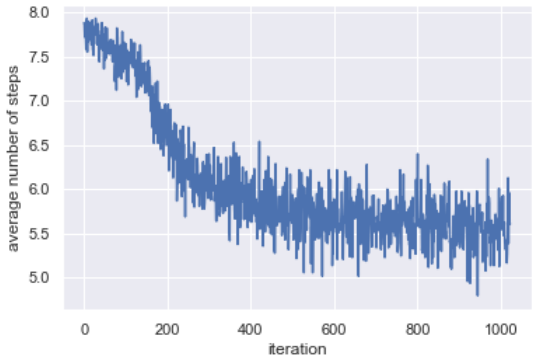


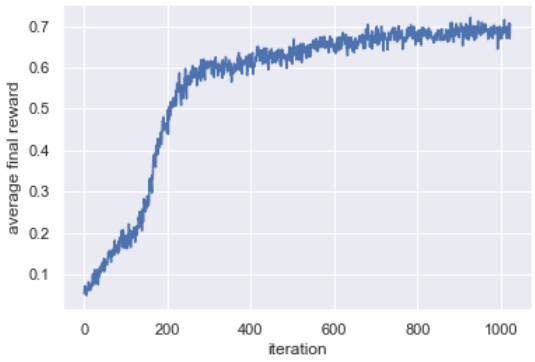
3. Positive rewards

Experiment: stop criteria = 0.75

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

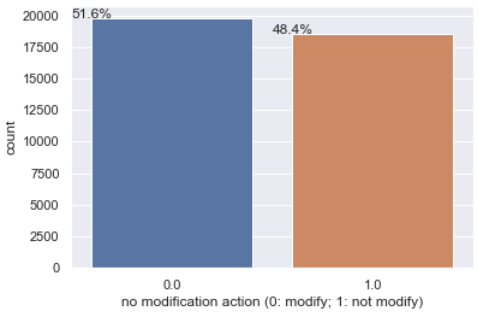


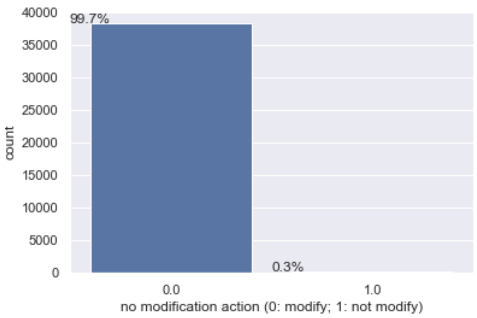


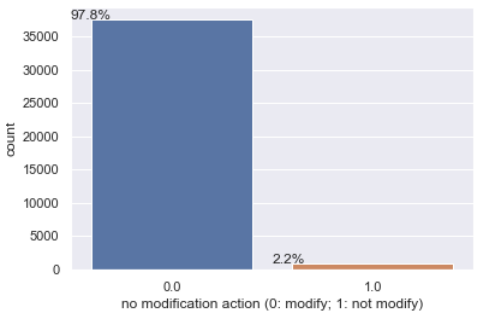


1. "No modification" action analysis

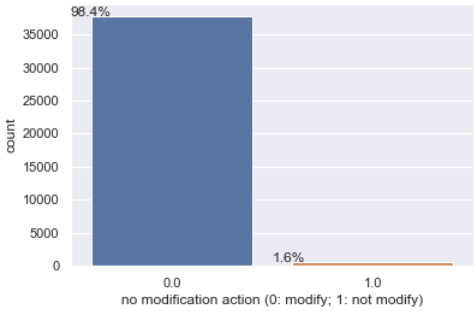
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

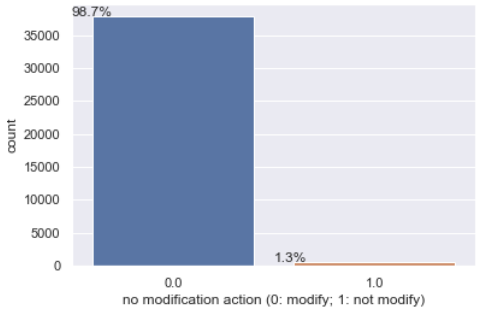




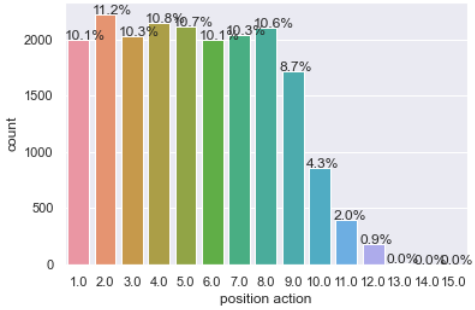


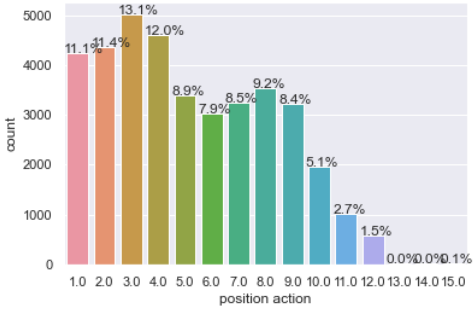
(10 iterations at 3/4) (last 10 iterations)

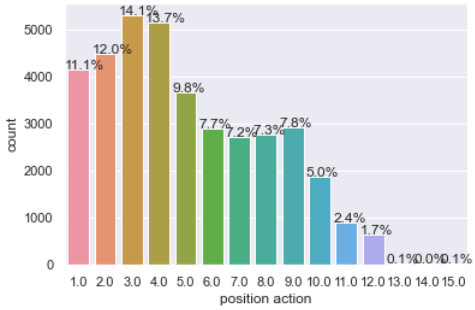


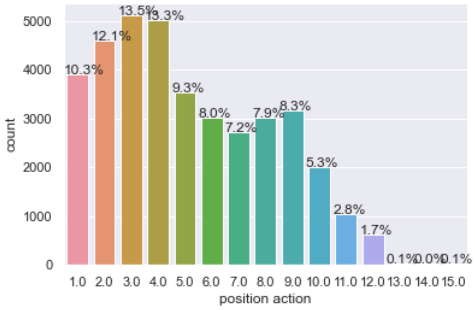


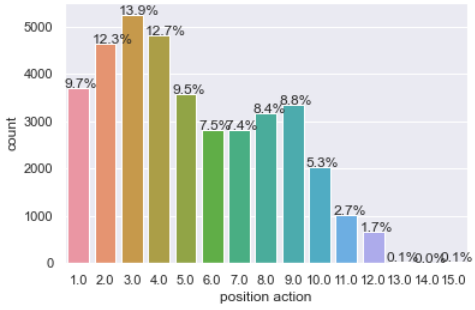
2. Position action analysis



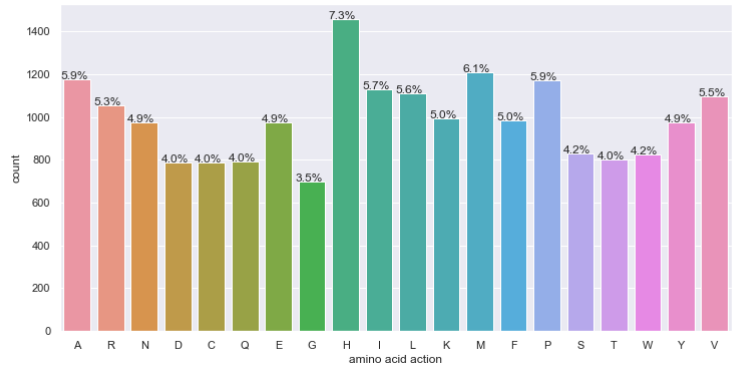


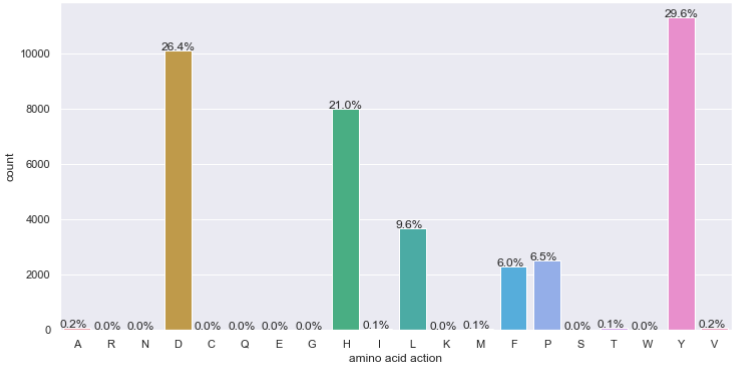


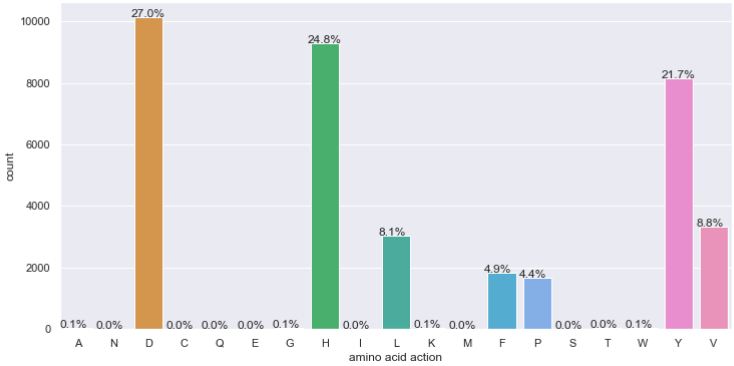


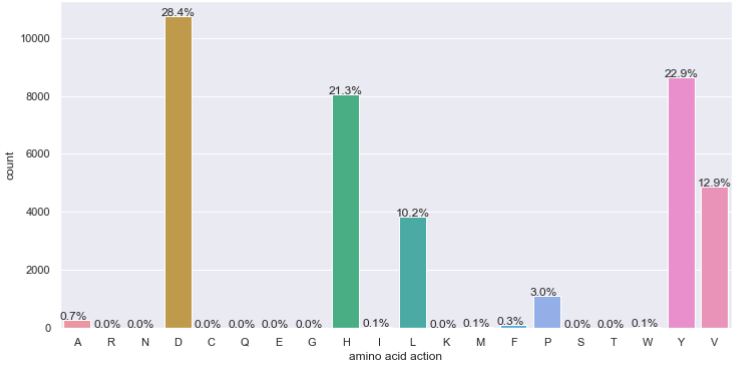


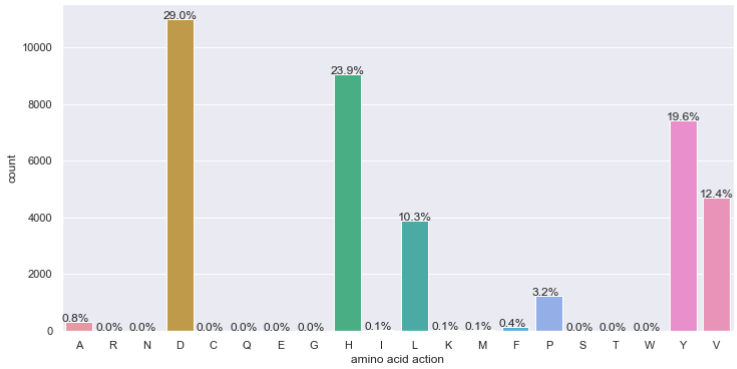
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.9% 5.3% 4.9% 4.0% 4.0% 4.0% 4.9% 3.5% 7.3% 5.7% 5.6% 5.0% 6.1% 5.0% 5.9% 4.2% 4.0% 4.2% 4.9% 5.5%

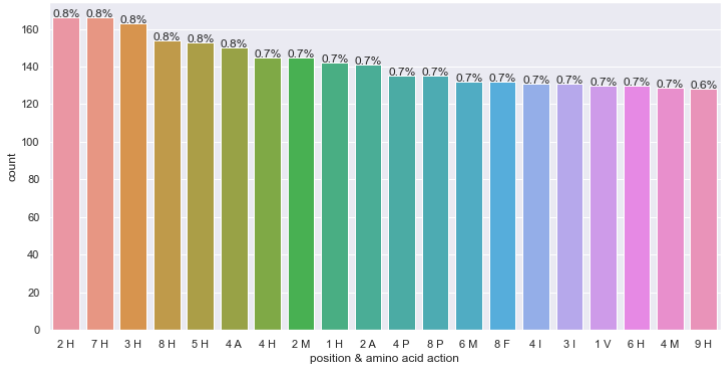

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.2% 0.0% 0.0% 26.4% 0.0% 0.0% 0.0% 0.0% 21.0% 0.1% 9.6% 0.0% 0.1% 6.0% 6.5% 0.0% 0.1% 0.0% 29.6% 0.2%

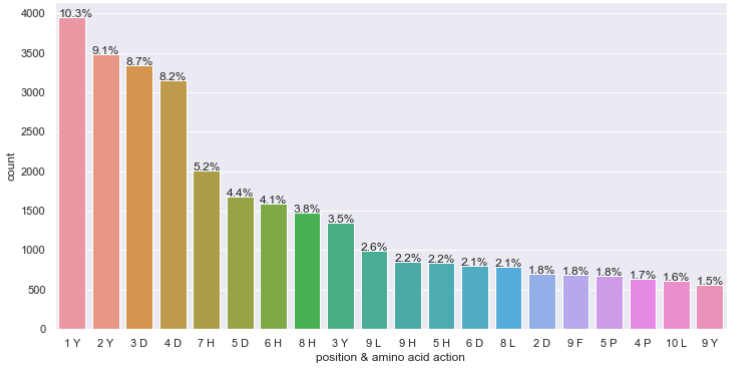

count
amino acid action
A N D C Q E G H I L K M F P S T W Y V
0.1% 0.0% 27.0% 0.0% 0.0% 0.0% 0.1% 24.8% 0.0% 8.1% 0.1% 0.0% 4.9% 4.4% 0.0% 0.0% 0.1% 21.7% 8.8%

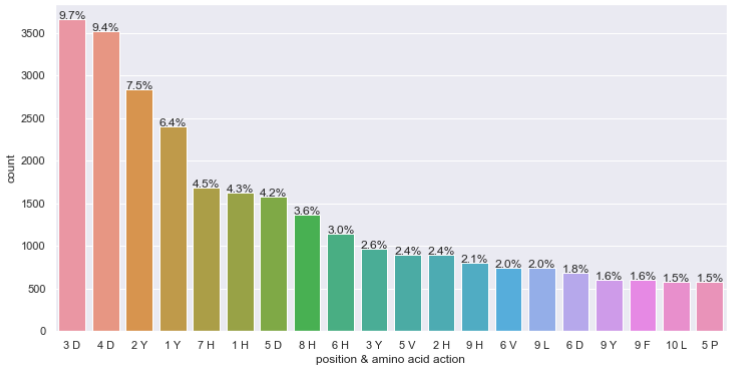

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.7% 0.0% 0.0% 28.4% 0.0% 0.0% 0.0% 0.0% 21.3% 0.1% 10.2% 0.0% 0.1% 0.3% 3.0% 0.0% 0.0% 0.1% 22.9% 12.9%

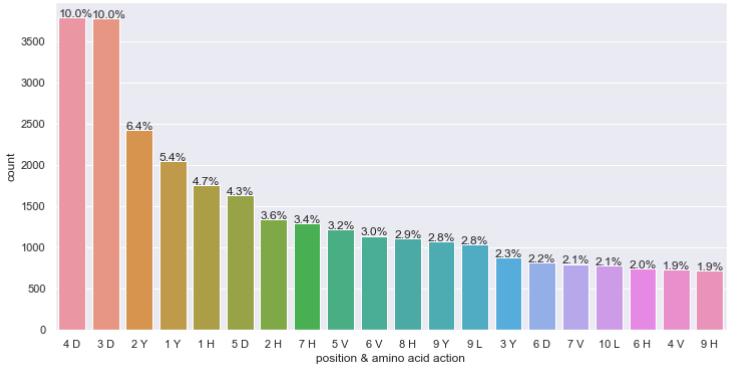


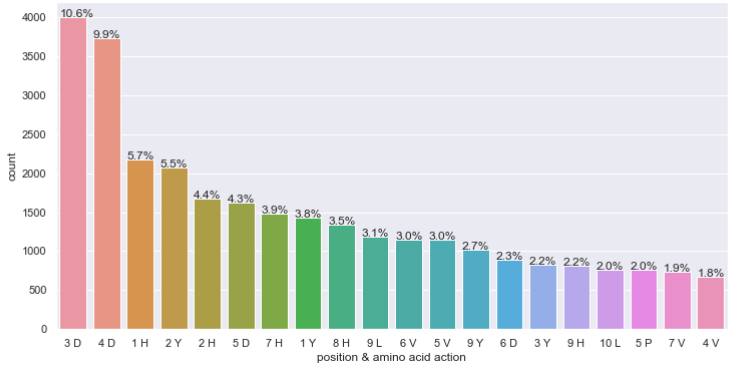
4. Position & Amino action analysis



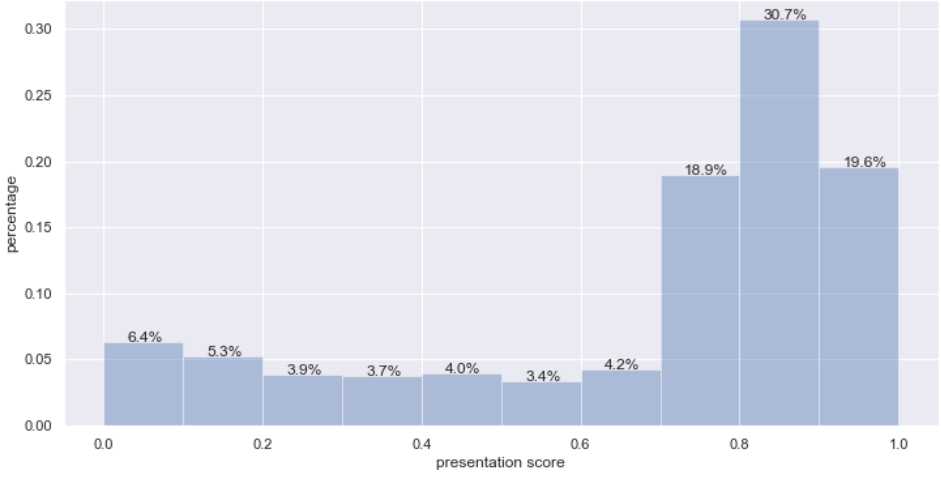




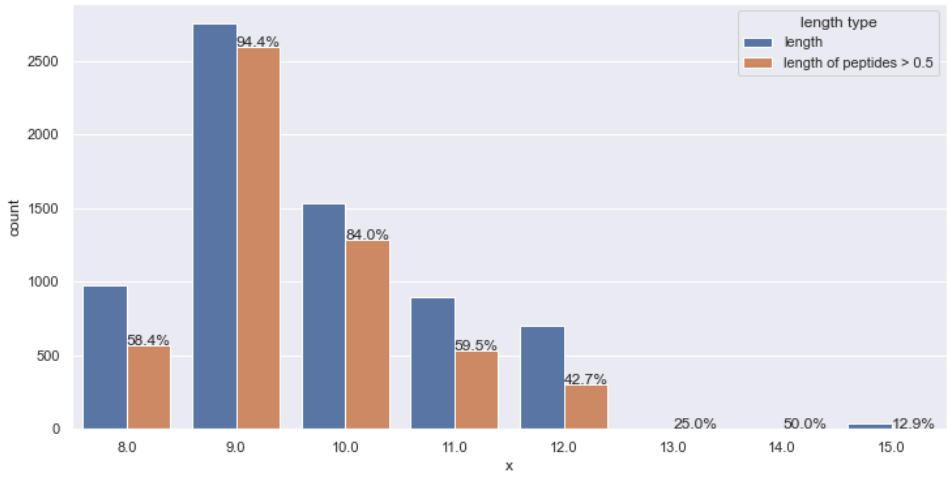




5. Reward analysis

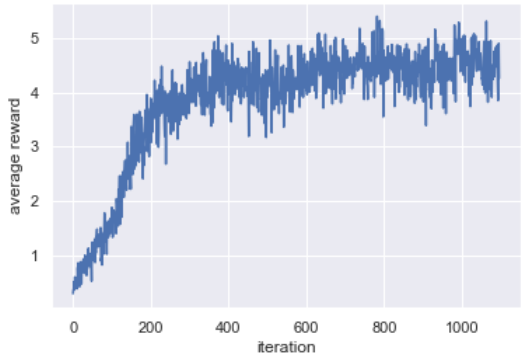


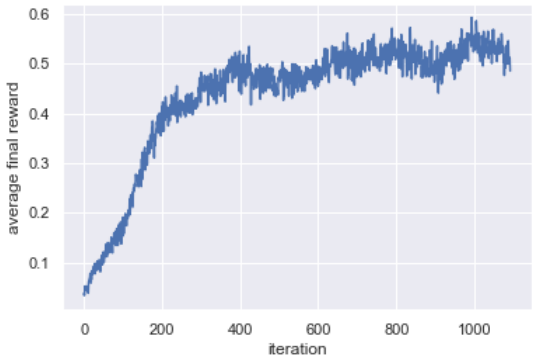
6. Length analysis



Experiment: stop criteria = 1.0

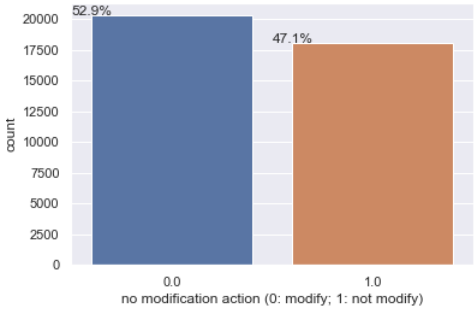
Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

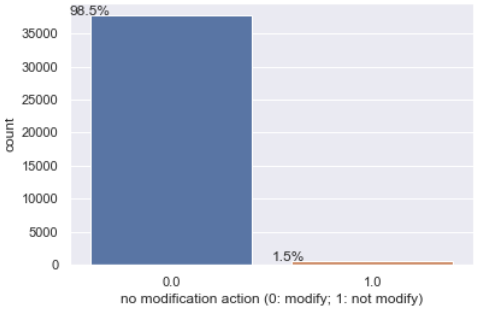


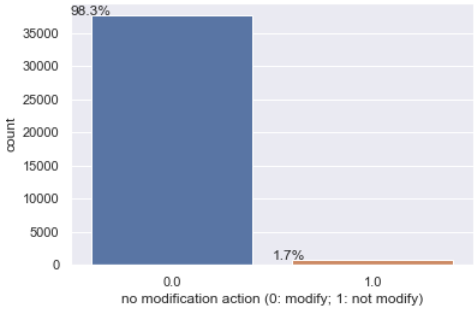


1. "No modification" action analysis

(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

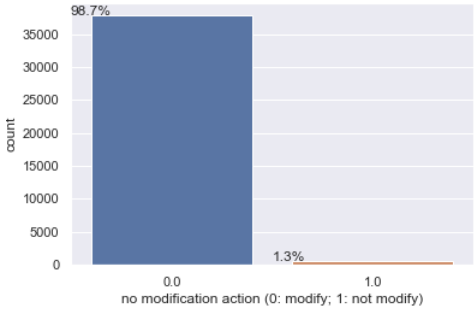


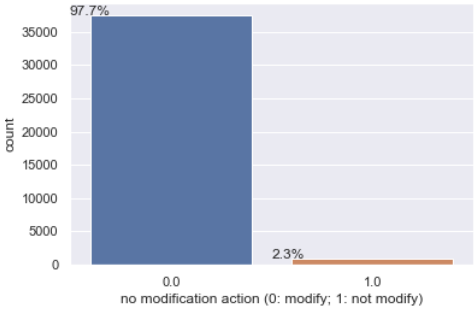




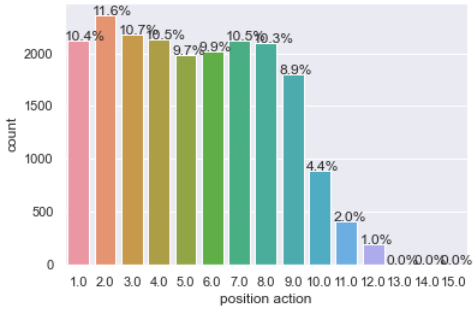
(10 iterations at 3/4)

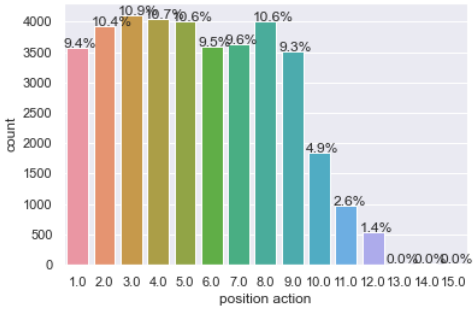
(last 10 iterations)

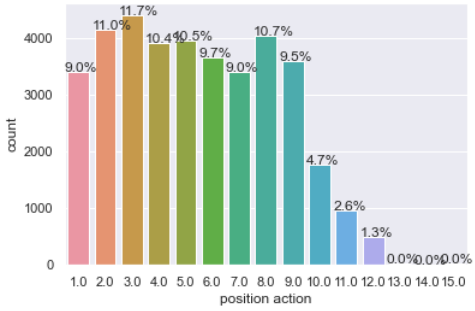


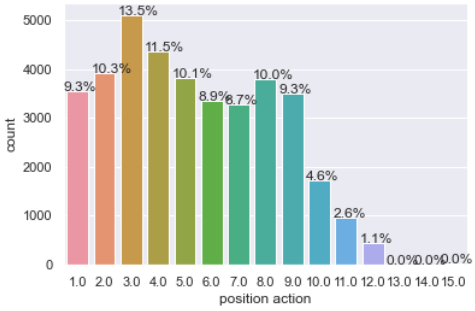


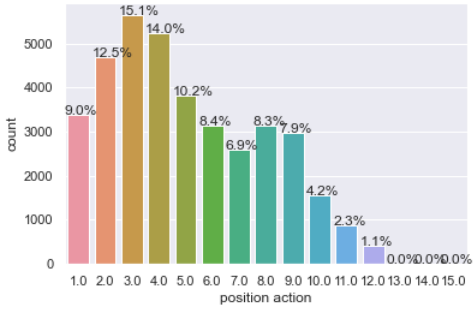
2. Position action analysis



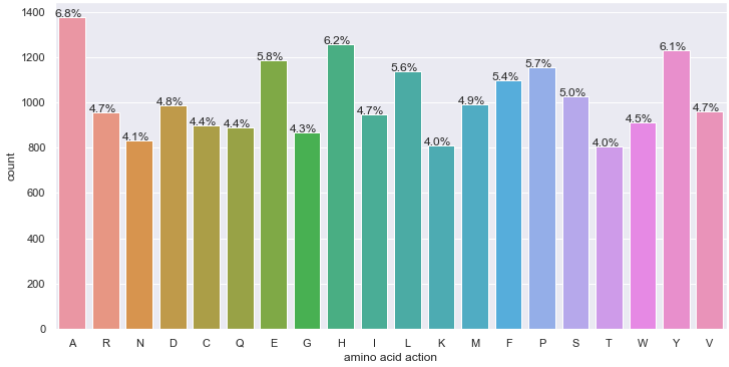


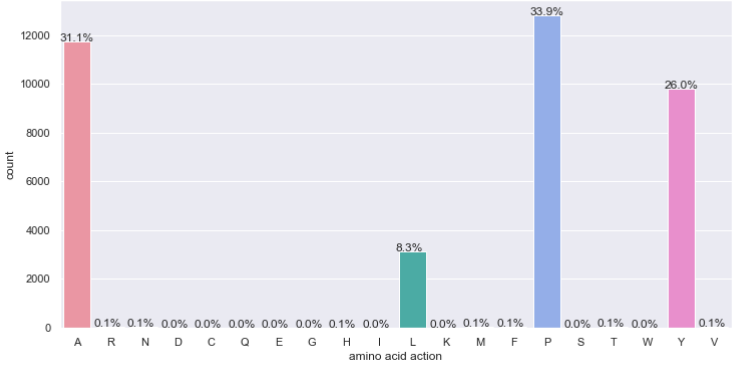


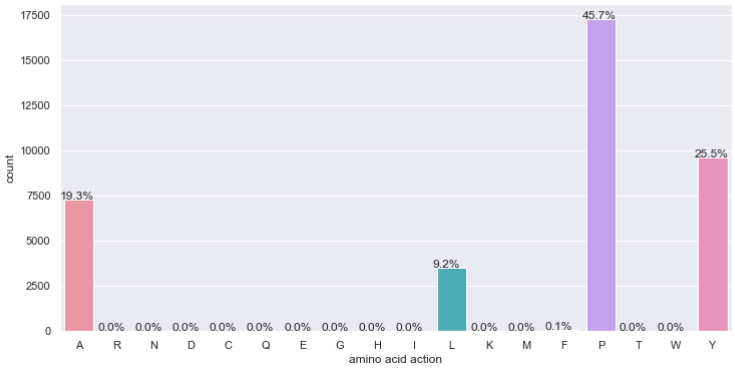


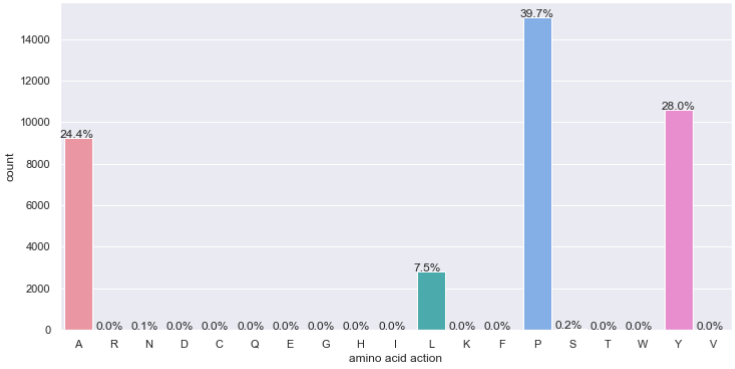


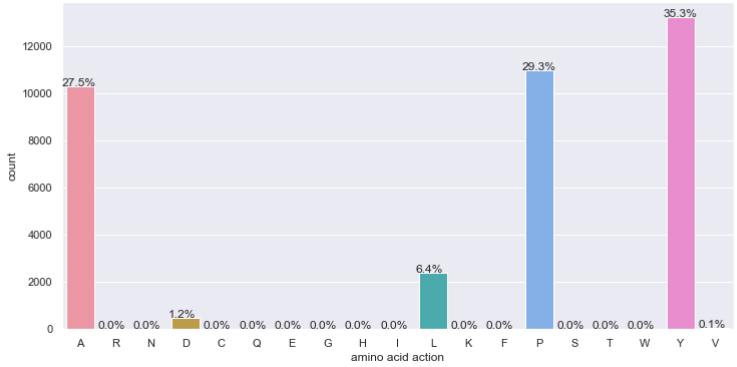
3. Amino action analysis



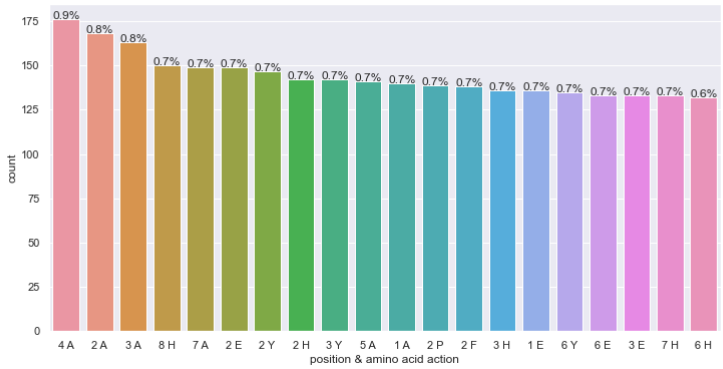


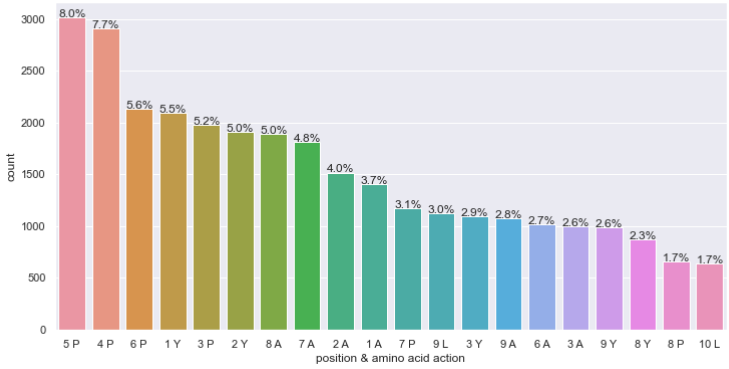


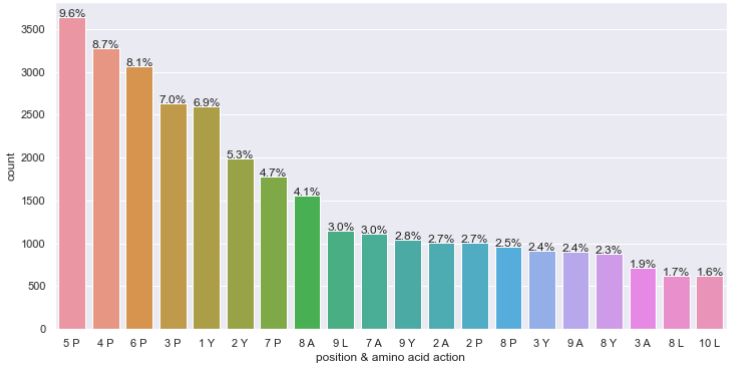


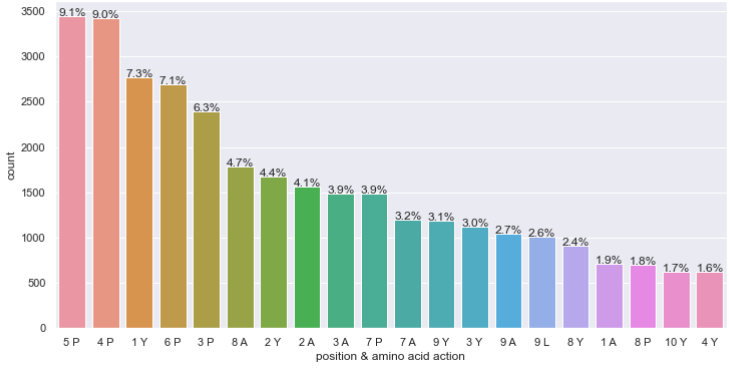


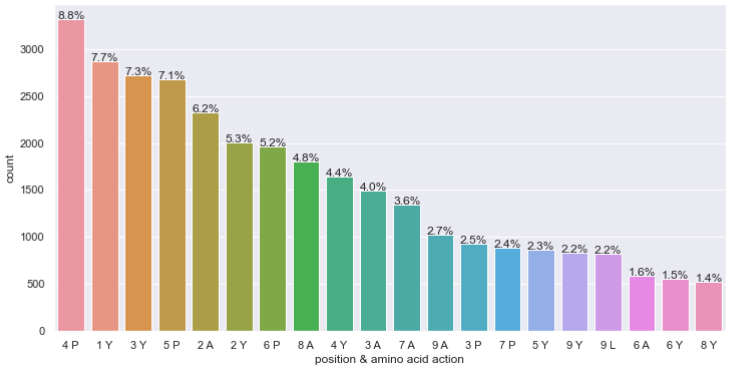
4. Position & Amino action analysis


count
position & amino acid action
0.9% 0.8% 0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.6%
4 A 2 A 3 A 8 H 7 A 2 E 2 Y 2 H 3 Y 5 A 1 A 2 P 2 F 3 H 1 E 6 Y 6 E 3 E 7 H 6 H


count
position & amino acid action
8.0% 7.7% 5.6% 5.5% 5.2% 5.0% 5.0% 4.8% 4.0% 3.7% 3.1% 3.0% 2.9% 2.8% 2.7% 2.6% 2.6% 2.3% 1.7% 1.7%
5 P 4 P 6 P 1 Y 3 P 2 Y 8 A 7 A 2 A 1 A 7 P 9 L 3 Y 9 A 6 A 3 A 9 Y 8 Y 8 P 10 L

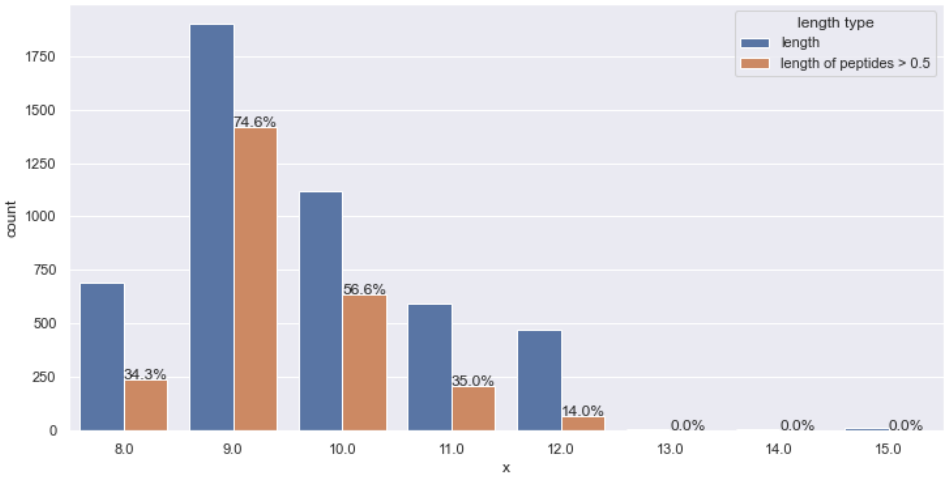

count
position & amino acid action
9.6% 8.7% 8.1% 7.0% 6.9% 5.3% 4.7% 4.1% 3.0% 3.0% 2.8% 2.7% 2.7% 2.5% 2.4% 2.4% 2.3% 1.9% 1.7% 1.6%
5 P 4 P 6 P 3 P 1 Y 2 Y 7 P 8 A 9 L 7 A 9 Y 2 A 2 P 8 P 3 Y 9 A 8 Y 3 A 8 L 10 L


count
position & amino acid action
9.1% 9.0% 7.3% 7.1% 6.3% 4.7% 4.4% 4.1% 3.9% 3.9% 3.2% 3.1% 3.0% 2.7% 2.6% 2.4% 1.9% 1.8% 1.7% 1.6%
5 P 4 P 1 Y 6 P 3 P 8 A 2 Y 2 A 3 A 7 P 7 A 9 Y 3 Y 9 A 9 L 8 Y 1 A 8 P 10 Y 4 Y



5. Reward analysis

6. Length analysis



Update (6-9)

Trick:

Add the number of steps as a feature to predict the actions $a \sim \pi(a|s,t)$ and the future rewards $Q(s,a)$. This trick is from the paper below [1].

Next step:

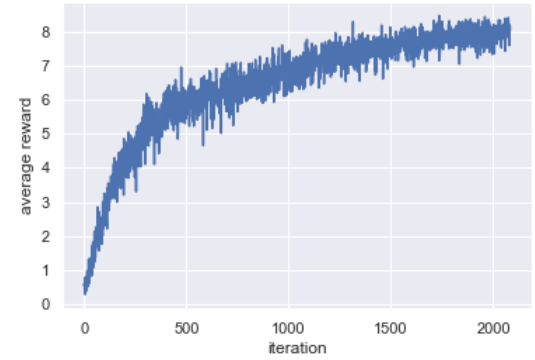
1. Add human peptide constraint when the data is available.
2. Compare the baselines (i.e., MCTS, motif, random) with the learned policy.
3. Write the paper
4. find alleles such as HLA-C with little training data, generate peptides for
   find alleles such as HLA-A with

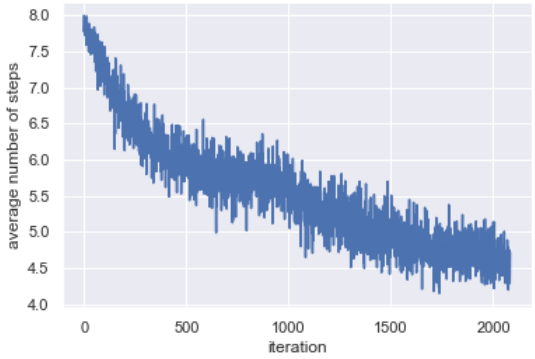
Idea:

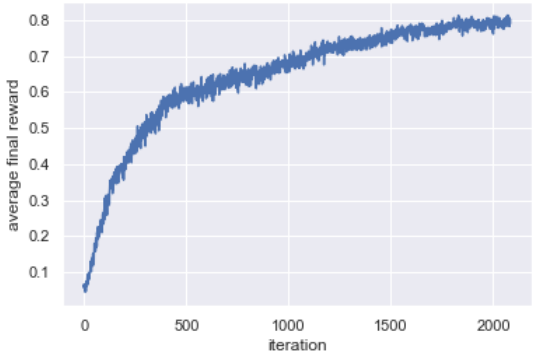
1. Combine MCTS with generative classifier

Experiment: stop criteria = 0.75

Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

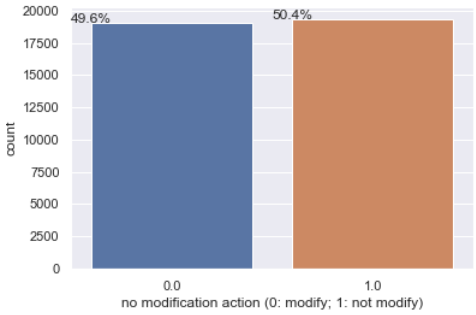


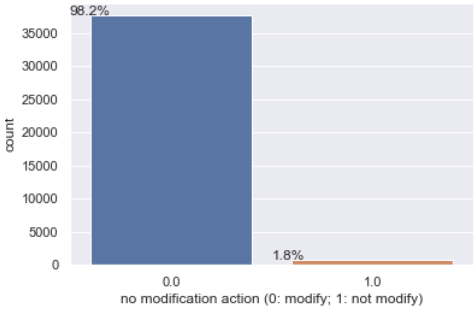


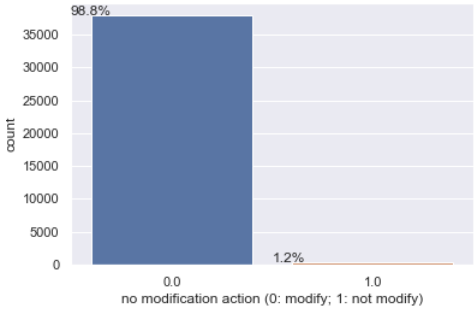


1. "No modification" action analysis

   (first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

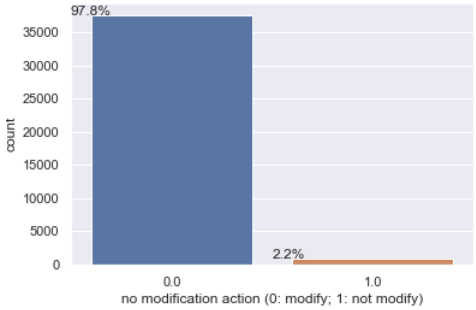


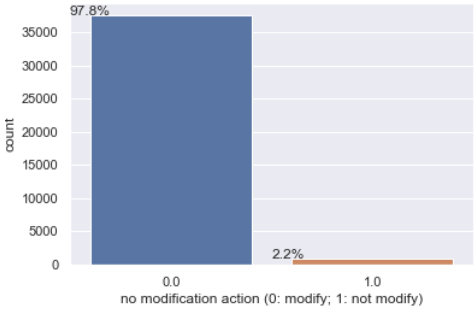




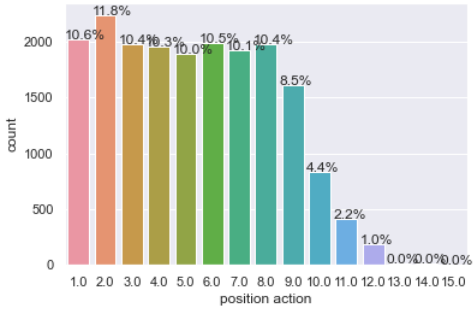
(10 iterations at 3/4)

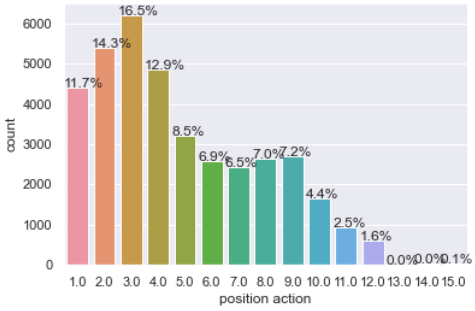
(last 10 iterations)

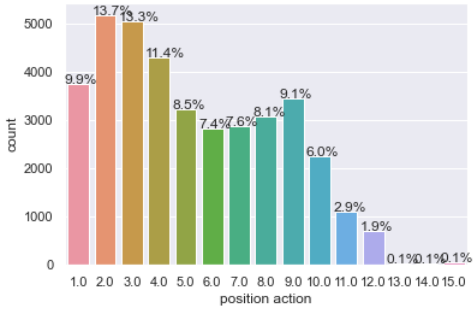


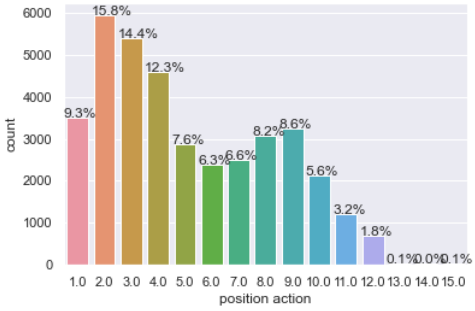


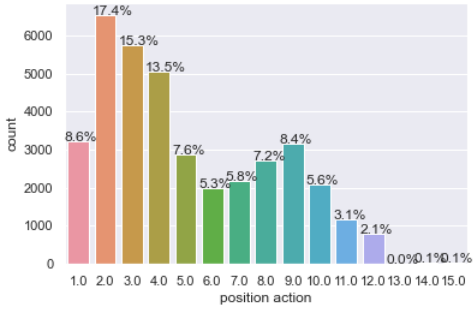
2. Position action analysis



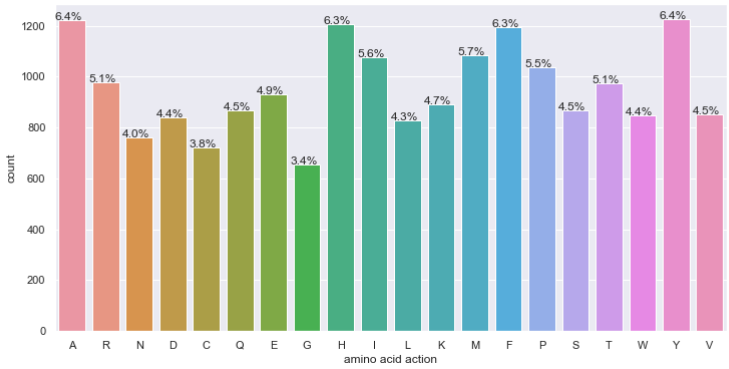


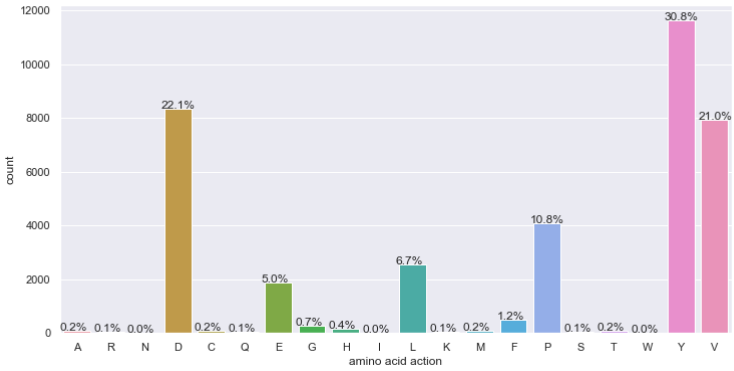


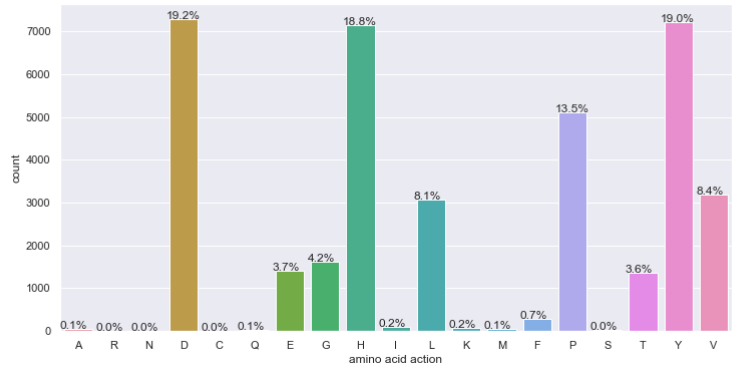


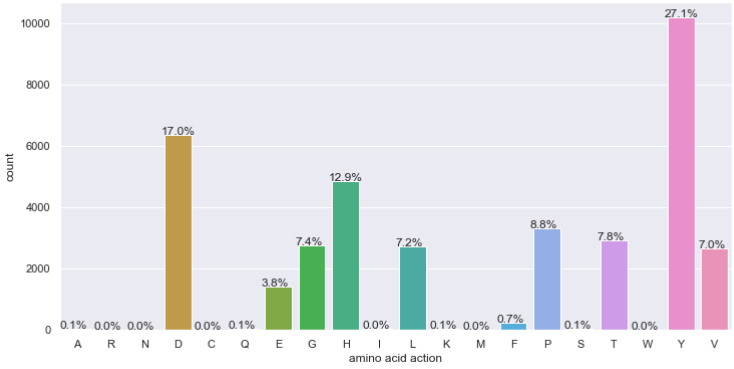


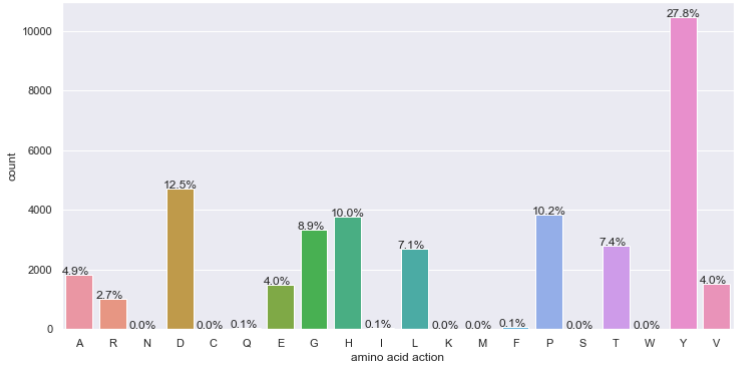
3. Amino action analysis



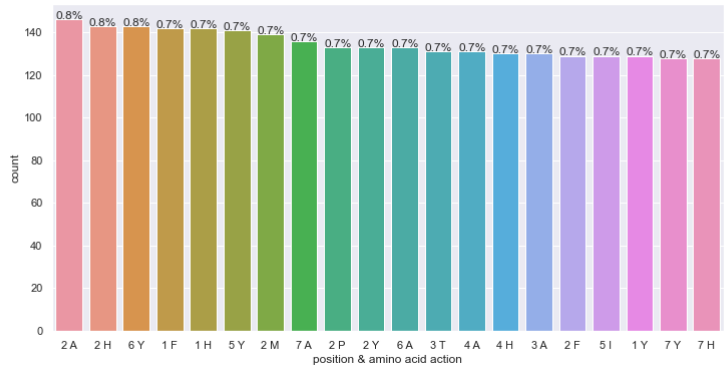


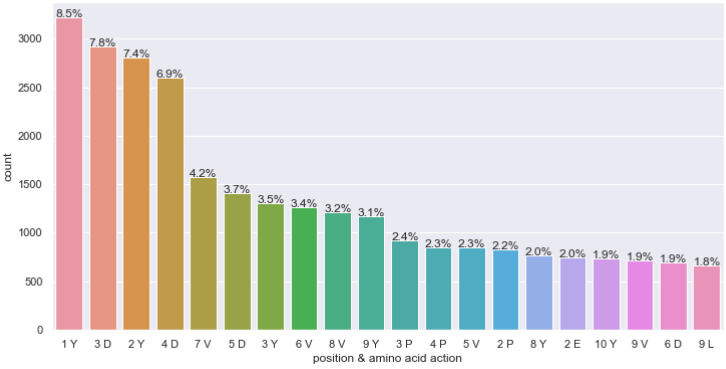


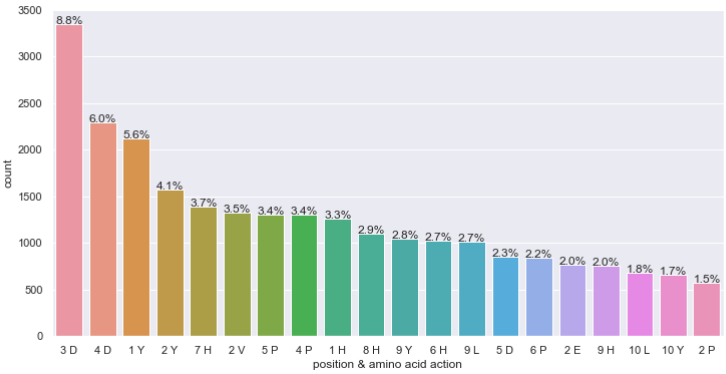


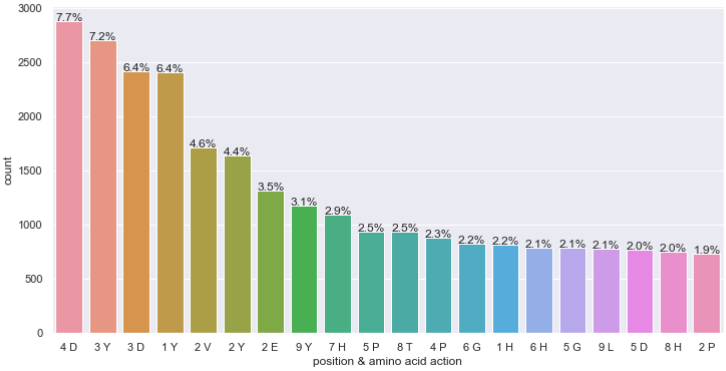


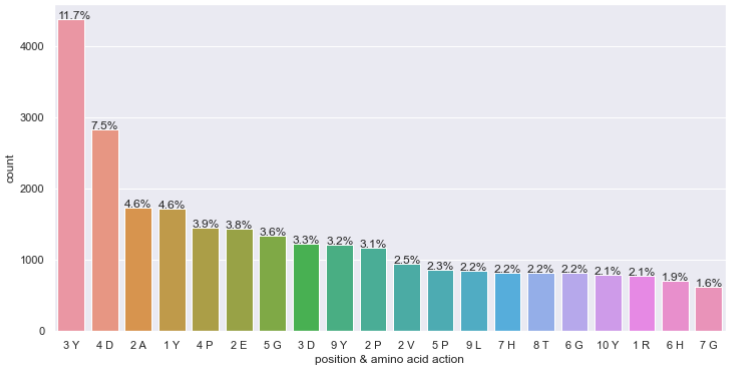
4. Position & Amino action analysis


count
0
20
40
60
80
100
120
140
0.8% 0.8% 0.8% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7% 0.7%
2 A 2 H 6 Y 1 F 1 H 5 Y 2 M 7 A 2 P 2 Y 6 A 3 T 4 A 4 H 3 A 2 F 5 I 1 Y 7 Y 7 H
position & amino acid action


count
0
500
1000
1500
2000
2500
3000
8.5% 7.8% 7.4% 6.9% 4.2% 3.7% 3.5% 3.4% 3.2% 3.1% 2.4% 2.3% 2.3% 2.2% 2.0% 2.0% 1.9% 1.9% 1.9% 1.8%
1 Y 3 D 2 Y 4 D 7 V 5 D 3 Y 6 V 8 V 9 Y 3 P 4 P 5 V 2 P 8 Y 2 E 10 Y 9 V 6 D 9 L
position & amino acid action

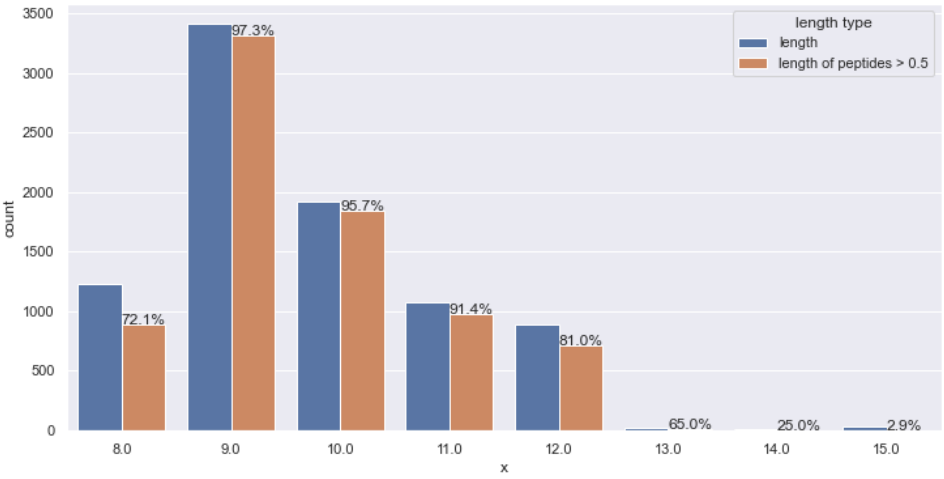

count
0
500
1000
1500
2000
2500
3000
3500
8.8% 6.0% 5.6% 4.1% 3.7% 3.5% 3.4% 3.4% 3.3% 2.9% 2.8% 2.7% 2.7% 2.3% 2.2% 2.0% 2.0% 1.8% 1.7% 1.5%
3 D 4 D 1 Y 2 Y 7 H 2 V 5 P 4 P 1 H 8 H 9 Y 6 H 9 L 5 D 6 P 2 E 9 H 10 L 10 Y 2 P
position & amino acid action


count
0
500
1000
1500
2000
2500
3000
7.7% 7.2% 6.4% 6.4% 4.6% 4.4% 3.5% 3.1% 2.9% 2.5% 2.5% 2.3% 2.2% 2.2% 2.1% 2.1% 2.1% 2.0% 2.0% 1.9%
4 D 3 Y 3 D 1 Y 2 V 2 Y 2 E 9 Y 7 H 5 P 8 T 4 P 6 G 1 H 6 H 5 G 9 L 5 D 8 H 2 P
position & amino acid action



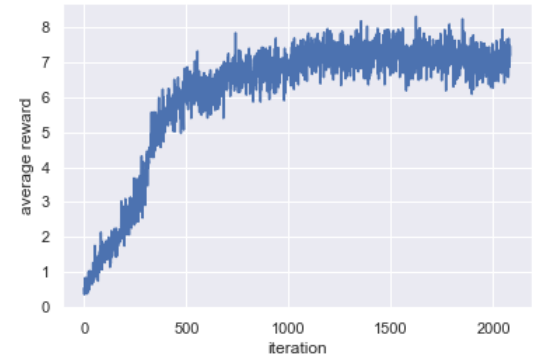
5. Reward analysis

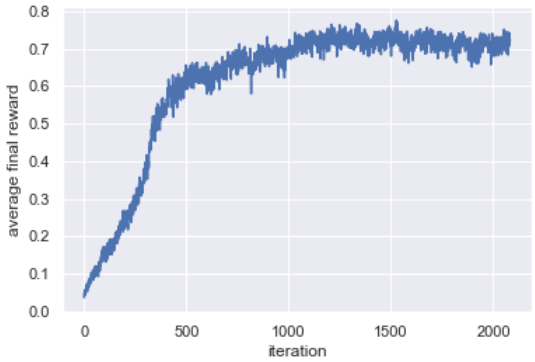
6. Length analysis



Experiment: no stopping criteria

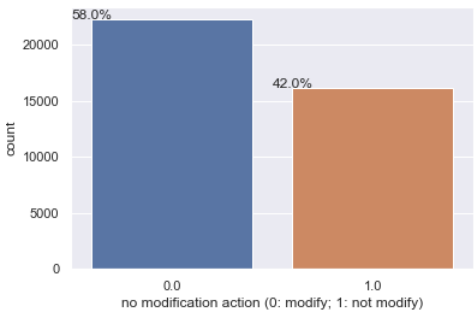
Setting 1 (only final rewards; sample rate = 50% from IEDB and 50% from random)

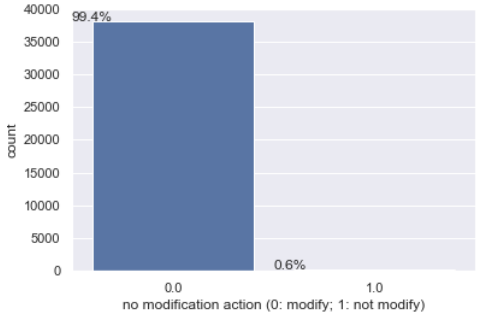


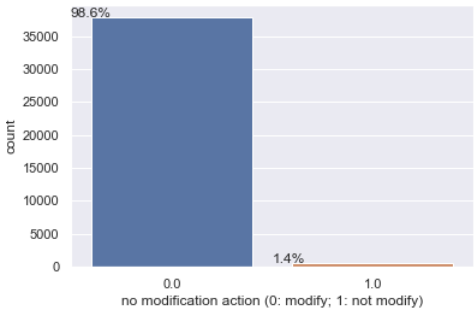


1. "No modification" action analysis

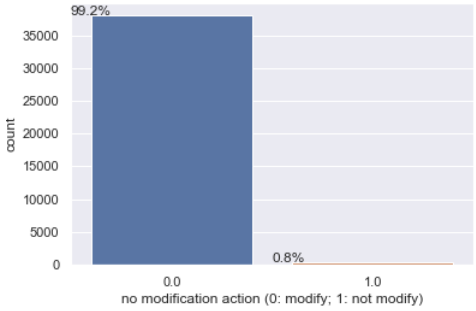
(first 10 iterations) (10 iterations at 1/4) (10 iterations in the middle)

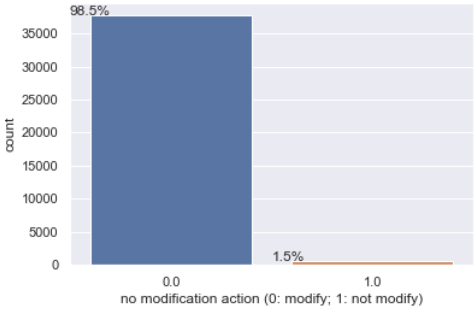




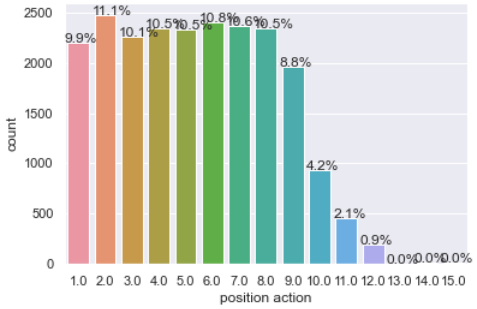


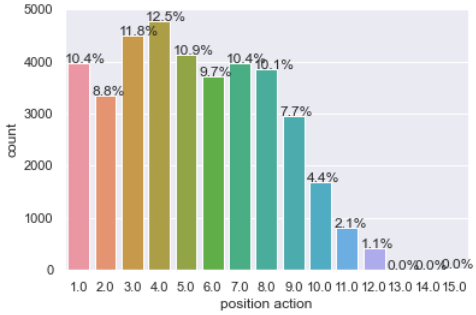
(10 iterations at 3/4) (last 10 iterations)

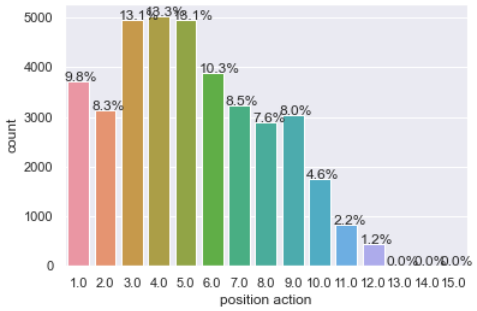


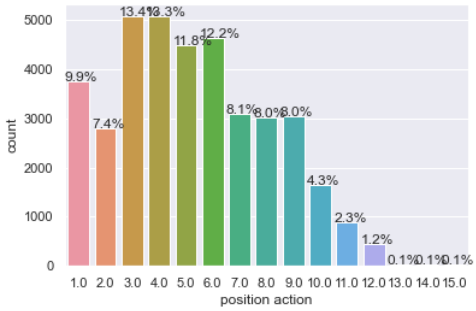


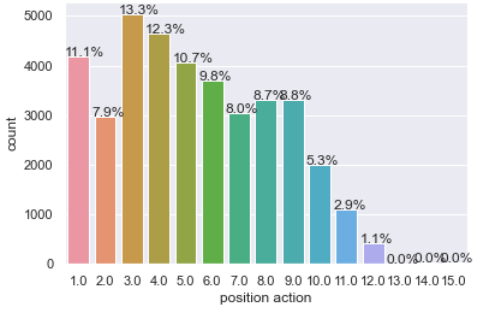
2. Position action analysis



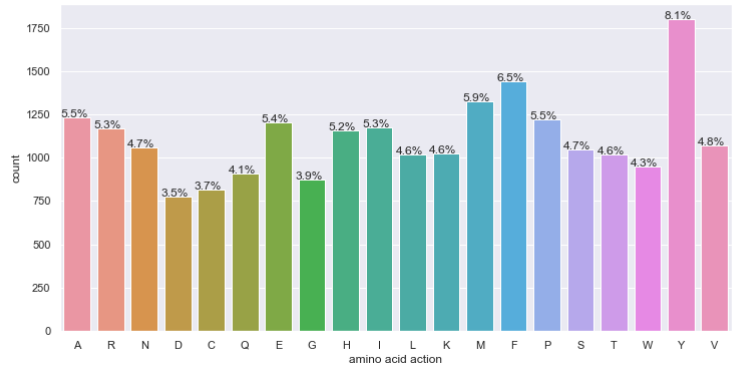


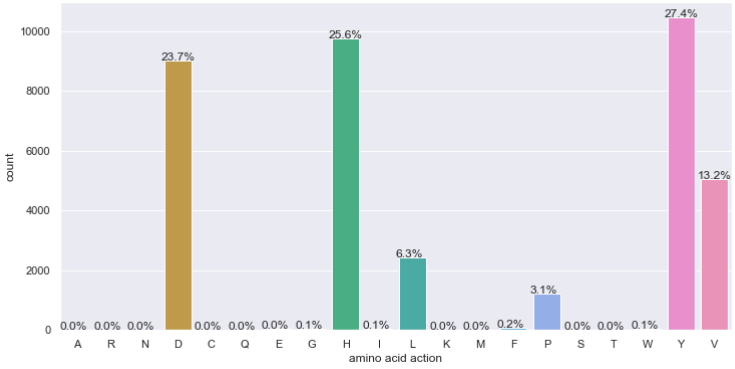


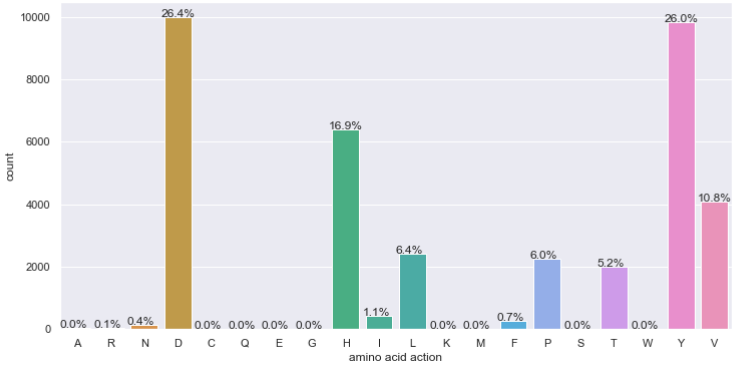


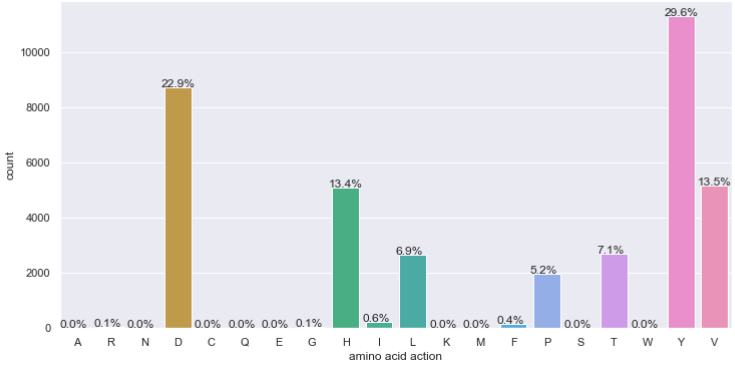


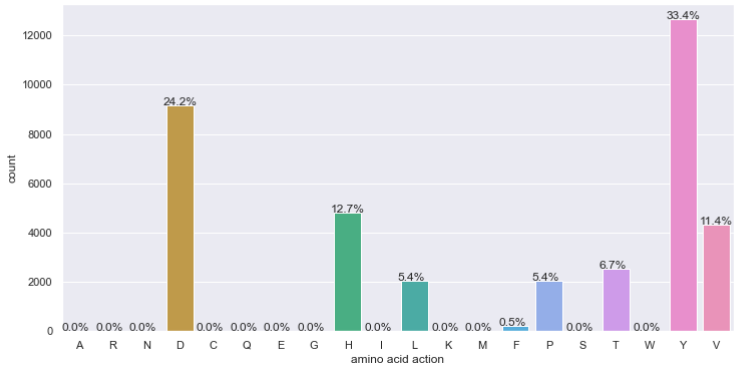
3. Amino action analysis


count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
5.5% 5.3% 4.7% 3.5% 3.7% 4.1% 5.4% 3.9% 5.2% 5.3% 4.6% 4.6% 5.9% 6.5% 5.5% 4.7% 4.6% 4.3% 8.1% 4.8%

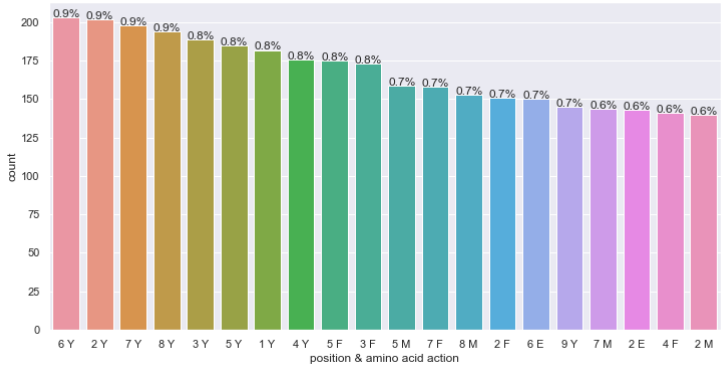

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.0% 0.0% 23.7% 0.0% 0.0% 0.0% 0.1% 25.6% 0.1% 6.3% 0.0% 0.0% 0.2% 3.1% 0.0% 0.0% 0.1% 27.4% 13.2%

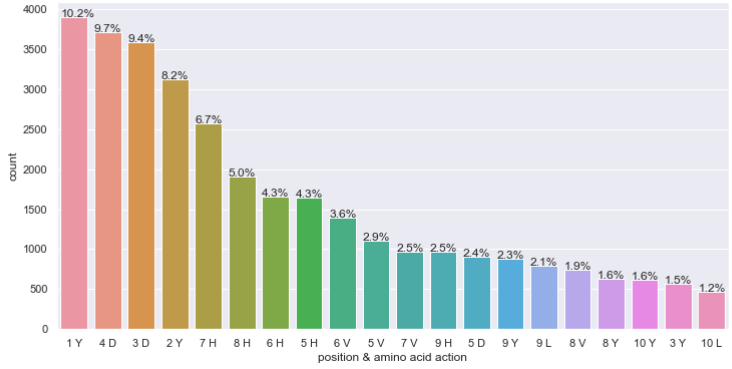

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.1% 0.4% 26.4% 0.0% 0.0% 0.0% 0.0% 16.9% 1.1% 6.4% 0.0% 0.0% 0.7% 6.0% 0.0% 5.2% 0.0% 26.0% 10.8%

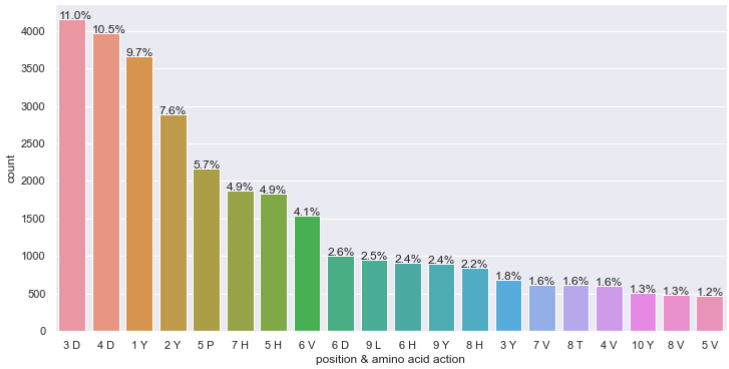

count
amino acid action
A R N D C Q E G H I L K M F P S T W Y V
0.0% 0.1% 0.0% 22.9% 0.0% 0.0% 0.0% 0.1% 13.4% 0.6% 6.9% 0.0% 0.0% 0.4% 5.2% 0.0% 7.1% 0.0% 29.6% 13.5%

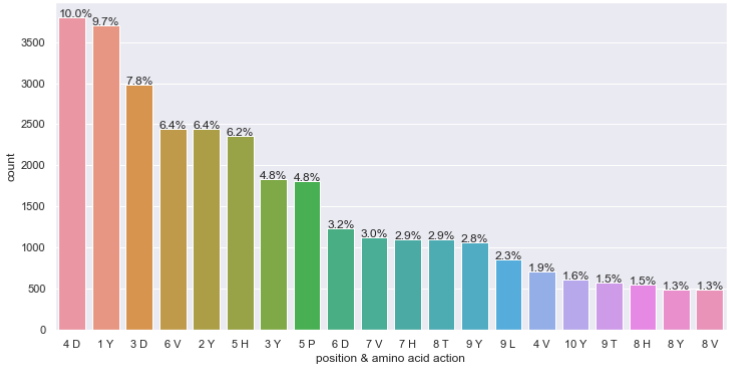


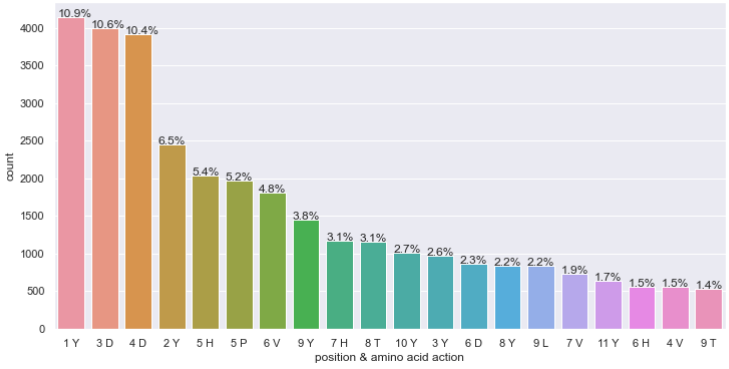
4. Position & Amino action analysis



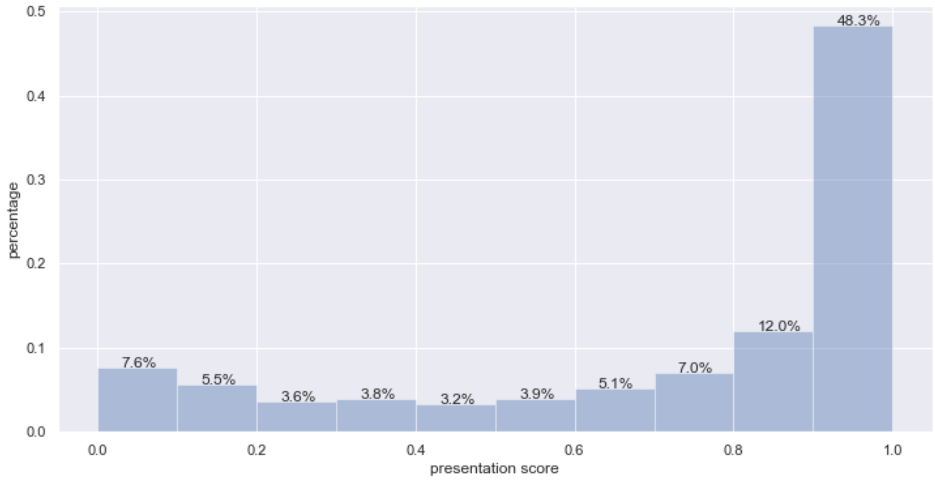




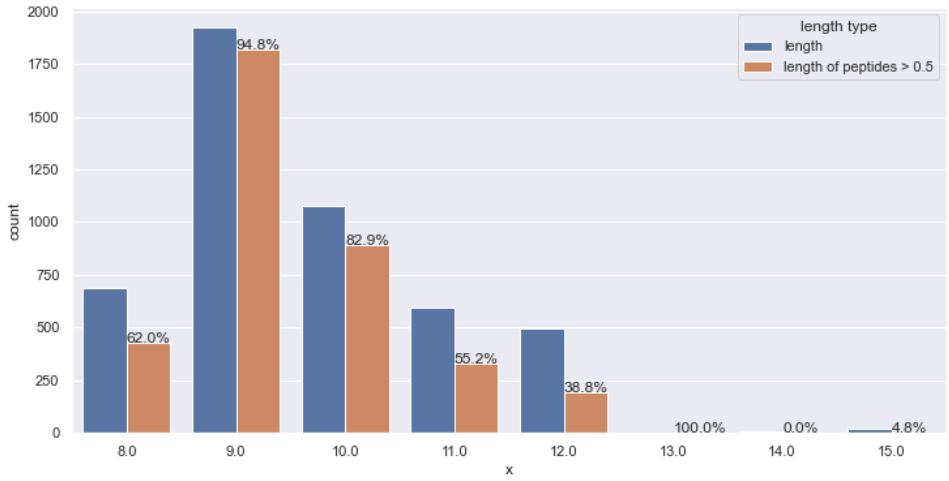




5. Reward analysis



6. Length analysis



Baseline 1: MCTS

important hyperparameters:

1. number of rollout for each allele
2. a criteria used to output good peptides
3. a tradeoff between exploitation and exploration within the UCB1 formula.

Algorithm detail:

- Each state is a sequence of amino acids. The root state is the empty sequence.
- For each state, we can pick one amino acid and get the next state by appending it to the sequence of this state.
    - We pick the amino acid that has the maximum value with the UCB1 formula

      $$a_k = \arg\max_a \frac{\sum_i r_i}{N(s,a)} + c_{puct} * \sqrt{\frac{2N(s)}{1+N(s,a)}}$$

      where c_{puct} is the tradeoff between exploitation and exploration; $N(s,a)$ is the visit count of state-action pair and $N(s)$ is the visit count of state s; r_i is the maximum representation score of i-th rollout going through the state-action pair.

    - The new state will be added to the tree.
- For each state s with sequence of length $l \geq 8$, we use *MHCflurry* to evaluate the presentation score of that sequence $p(s)$.
- The terminal state has the sequence of length $l = 15$.
- After reaching the terminal state, we backpropagate the presentation score $r_i^{t-1} = \max(r_i^t, p(s_i^{t-1}))$ in which $r_i^T = p(s_i^T)$.

Use the motif position to reorder the sequence

Question:

1. How to specify the number of rollouts?

In the RL model with stop criteria = 0.75, we have 3,688 alleles in total. Each allele is associated with about 400 episodes.

It could be unfair to set the number of rollouts to be 400. But I guess 5000 rollouts could be a good choice?

2. How to compare the MCTS and other baselines with the RL model?

Compare the space of generated peptides, e.g., t-sne plot.

For each allele, run baselines and our RL model for 1000 rounds.

Some metrics: number of optimized peptides (> 0.75); percentage of unique peptides; average pairwise edit distances between optimized peptides; overlap with training data;

t-sne plot

Some preliminary results:

(10,000 rollouts for each allele, $c_{puct} = 0.5$)

time cost for each allele: about 3 hours

```
Threshold = 0.5:
Number of peptides, allele name
   1325        HLA-B*57:01
    896        HLA-A*29:02
    610        HLA-A*24:02
    544        HLA-B*35:01
    534        HLA-B*44:02
    464        HLA-A*02:03
    435        HLA-A*02:01
    432        HLA-A*31:01
    420        HLA-A*02:07
    364        HLA-A*01:01
    362        HLA-A*03:01
    282        HLA-A*68:02
    175        HLA-B*51:01
    164        HLA-B*54:01
```

Threshold = 0.75:

554 HLA-B*57:01*

*412 HLA-A*29:02

280 HLA-A*24:02*

*274 HLA-B*35:01

266 HLA-B*44:02*

*258 HLA-A*02:03

223 HLA-A*02:01*

*182 HLA-A*02:07

177 HLA-A*31:01*

*153 HLA-A*03:01

138 HLA-A*01:01*

*127 HLA-A*68:02

79 HLA-B*51:01*

72 *HLA-B*54:01

Baseline 2: Position Weight Matrix (Motif)

Similar to the prior distribution we derived in the previous experiments.

Baseline 3: Random Policy

Update (6-23)

- An example of comparison among methods with different experimental settings:

Model	Max. Penalized logP	**Model**	
GVAE + BO (Kusner et al., 2017) [1]	2.87 ± 0.06	VAE	Exploitation
SD-VAE (Dai et al., 2018) [1]	3.50 ± 0.44	VAE	
CVAE + BO (Gómez-Bombarelli et al., 2018) [2]	4.85 ± 0.17	VAE	
ORGAN (Guimaraes et al., 2017) [1]	3.52 ± 0.08	GAN	
JT-VAE (Jin et al., 2018a) [1]	4.90 ± 0.33	VAE	
ChemTS (Yang et al., 2017)	5.6 ± 0.5	RNN	
GCPN (You et al., 2018) [1]	7.87 ± 0.07	PN + GAN	
Random SELFIES	6.19 ± 0.63	Random Search	Exploration
GB-GA (Jensen, 2019) [3]	7.4 ± 0.9	GA	
GB-GA (Jensen, 2019) [4]	$\mathbf{15.76 \pm 5.71}$	GA	
GA (here)	12.61 ± 0.81	GA	
GA + D (here)	13.31 ± 0.63	GA + DNN	
(**GA + D(t)** (here) [5]	$\mathbf{20.72 \pm 3.14}$)	GA + DNN	

[1] average of three best molecules after performing Bayesian optimization on the latent representation of a trained VAE
[2] two best molecules of a single run
[3] averaged over 10 runs with 20 molecules per generation with a molecular weight (excl. hydrogen) smaller than 39.15 ± 3.50 g/mol run for 50 generations
[4] averaged over 10 runs with 500 molecules up to 81 characters per generation and 100 generations
[5] averaged over 5 runs with 1000 generations each

- The baselines used in the paper "model-based RL for biological sequence design"

- **RegEvolution**: Local search based on regularized evolution (Real et al., 2019), which has performed well on other black-box optimization tasks and can be seen as an instance of directed evolution.
- **DbAs**: Cross-entropy optimization using variational autoencoders (Brookes & Listgarten, 2018).
- **FBGAN**: Cross entropy optimization using generative adversarial networks (Gupta & Zou, 2018).
- **Bayesopt GP**: Bayesian optimization using a Gaussian process regressor and activation maximization as acquisition function solver.
- **Bayesopt ENN** Bayesian optimization using an ensemble of neural network regressors and activation maximization as acquisition function solver.

activation maximization [2] simply means back propagation. Specifically, the discrete biological one-hot sequence vectors are preprocessed with continuous relaxation by adding a latent variable z. This

1. Comparison between MCTS and RL model

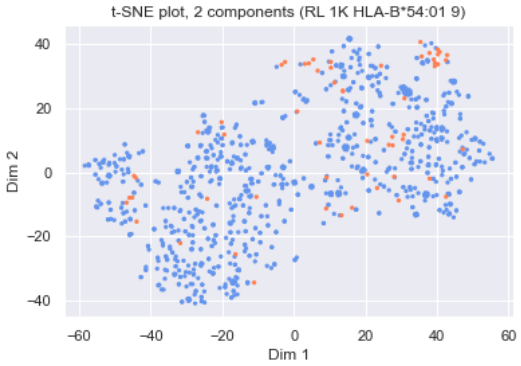
- calculate the distance between peptides of the same length.
- compare the overlapping peptides generated by the RL and generated by the MCTS.
- T-sne plot
- Use the same representation with the MHCflurry and MLP and fully connected layer. *Do not use the gradients to update the padding part.*

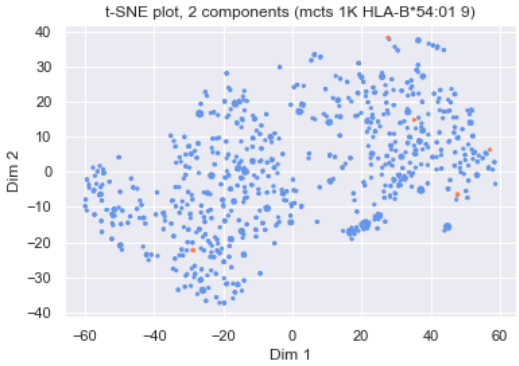
Advantages of the RL approach:

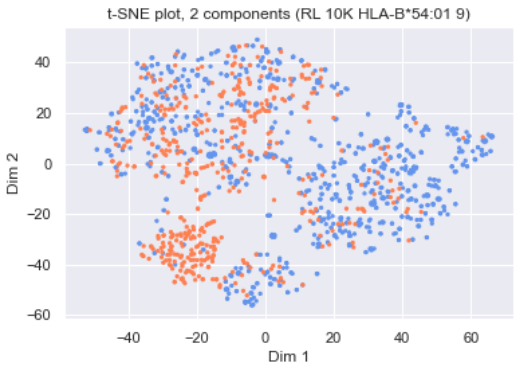
- Good performance
- Wider space
- Interpretability

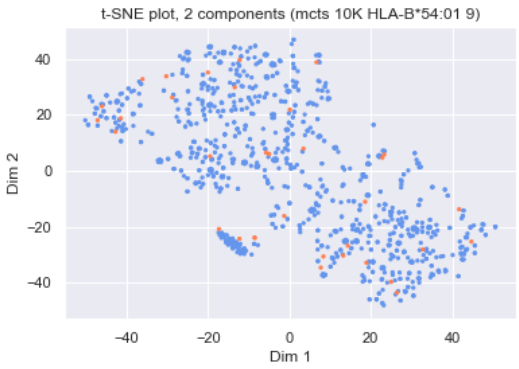
1.1 t-sne plot

- Allele: HLA-B*54:01 ; Length = 9

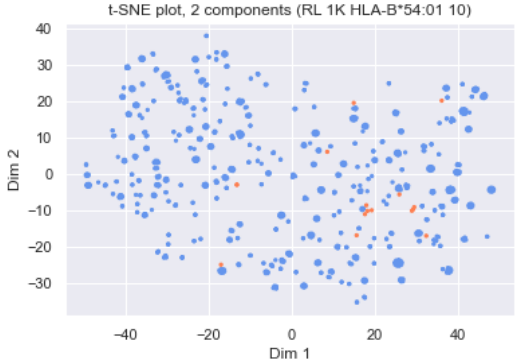




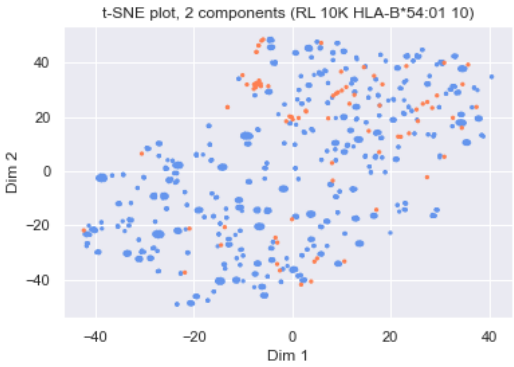


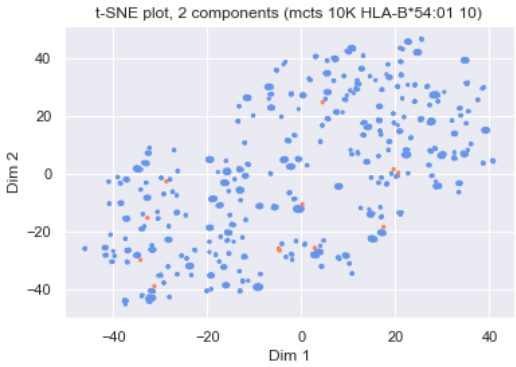


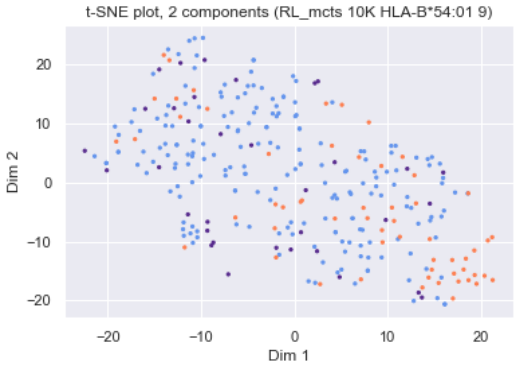
- Allele: HLA-B*54:01 ; Length = 10

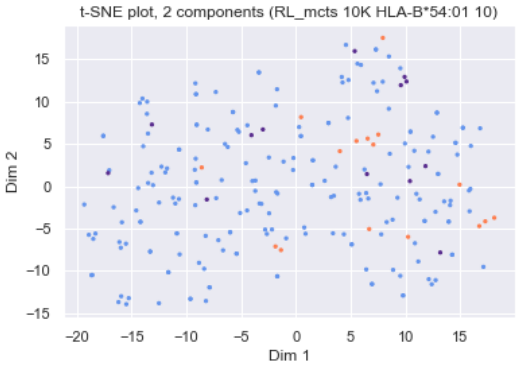


None

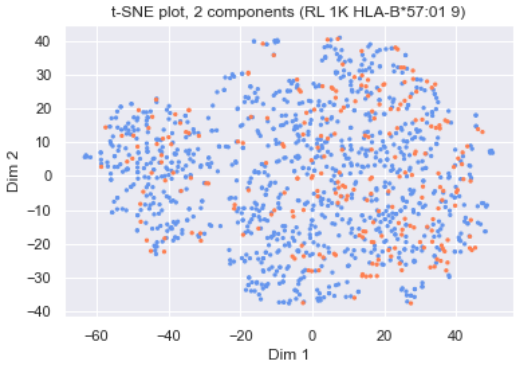


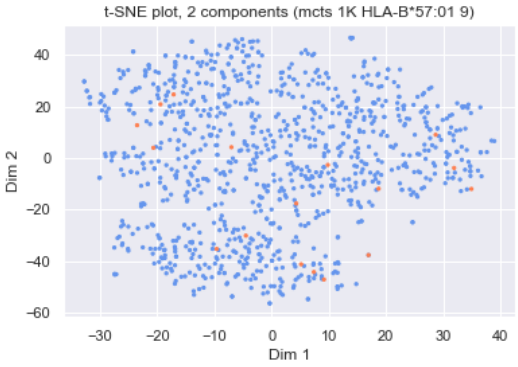


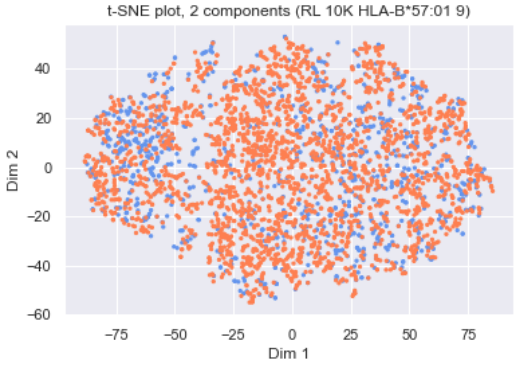


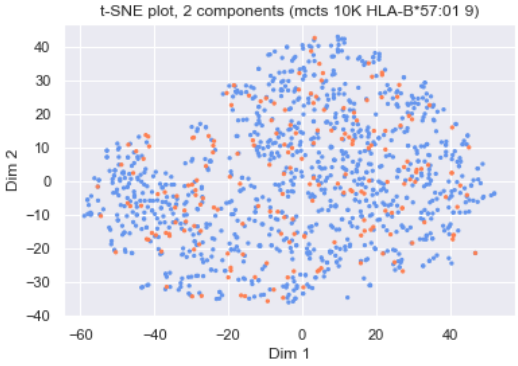


- Allele: HLA-B*57:01 ; Length = 9

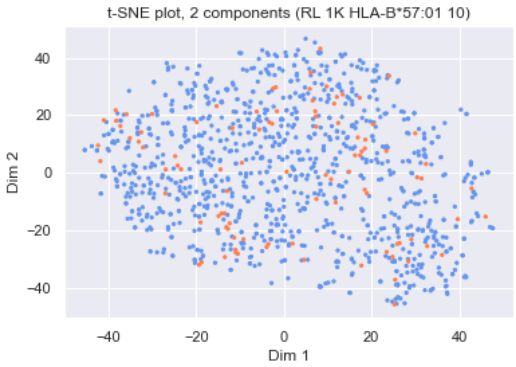


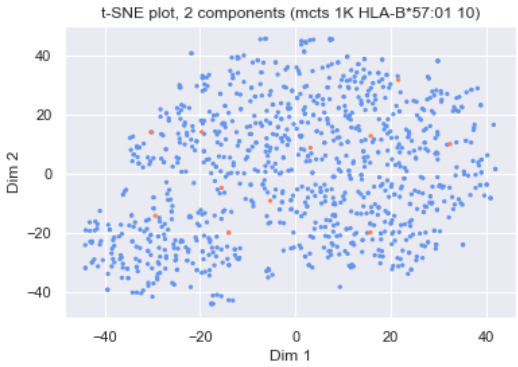


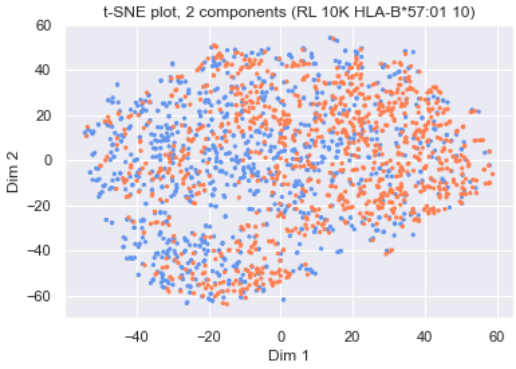


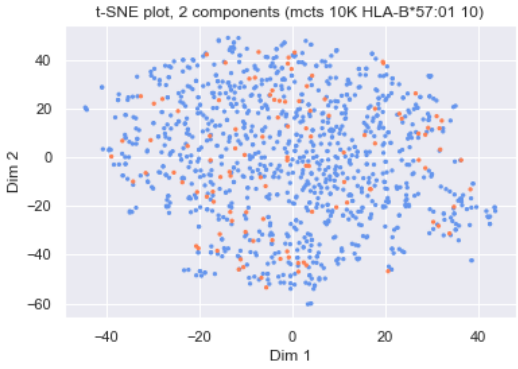


- Allele: HLA-B*57:01 ; Length = 10

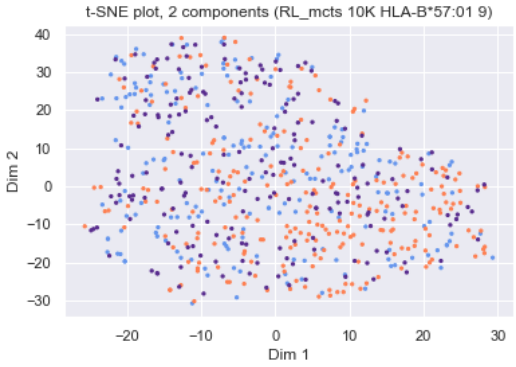


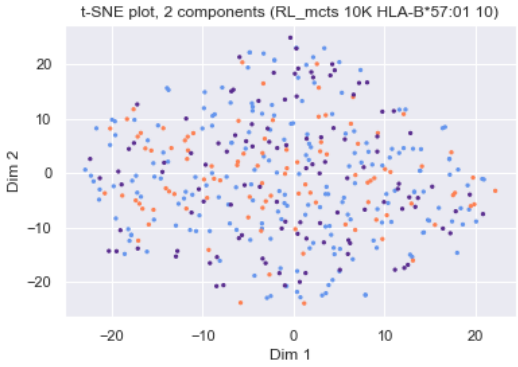




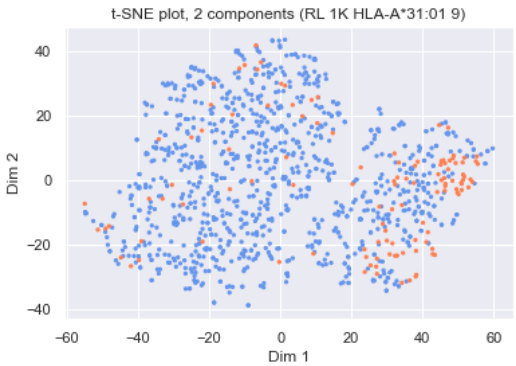


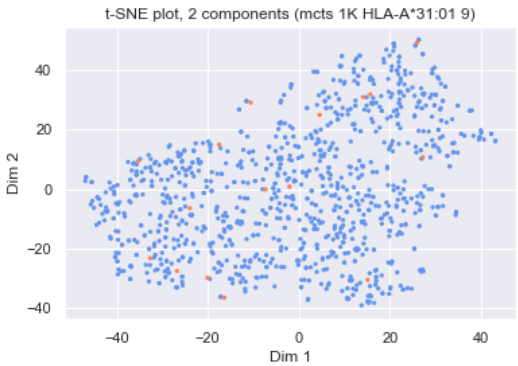
RL 1K vs MCTS 10K

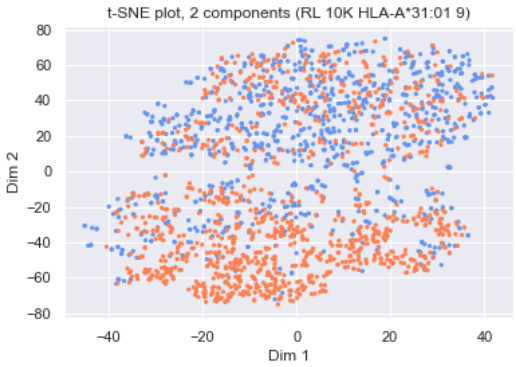


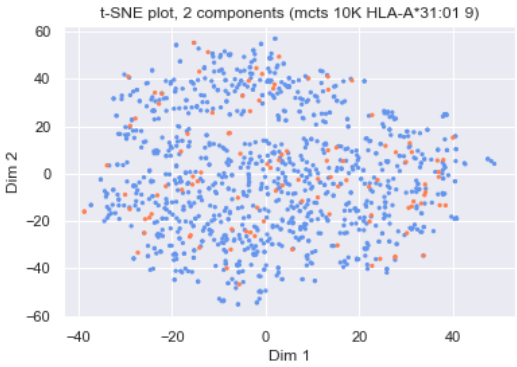


- Allele: HLA-A*31:01 ; Length = 9

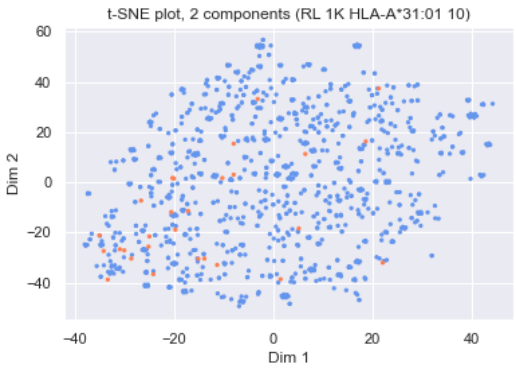


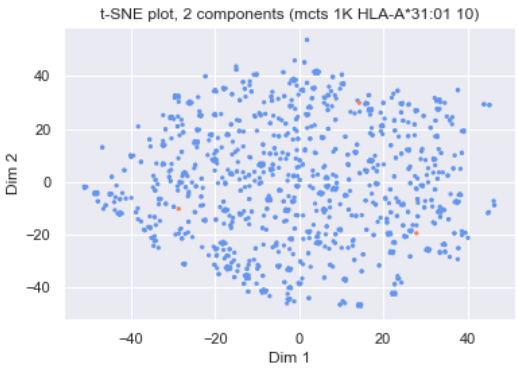


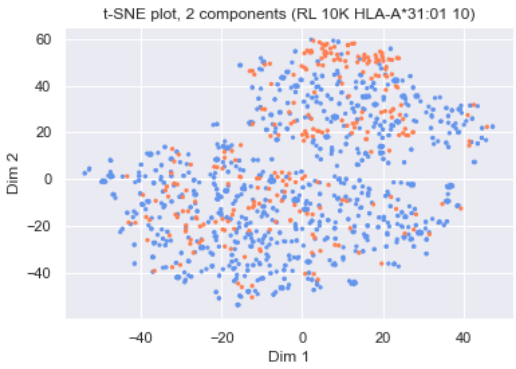


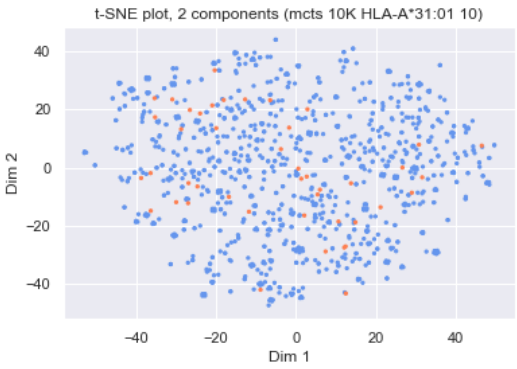


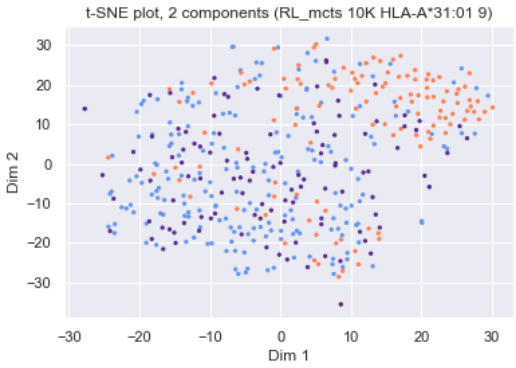
- Allele: HLA-A*31:01 ; Length = 10

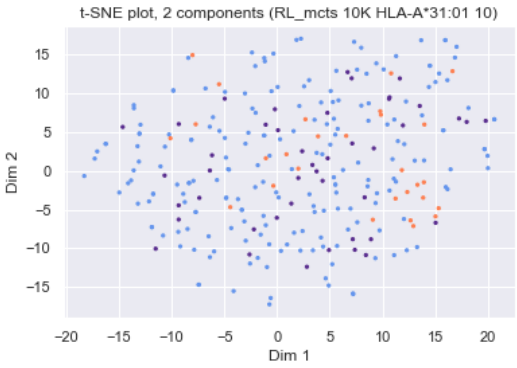






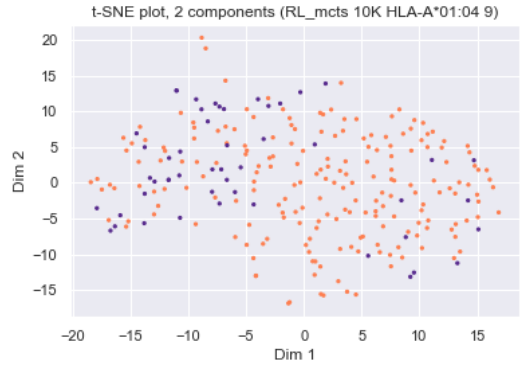


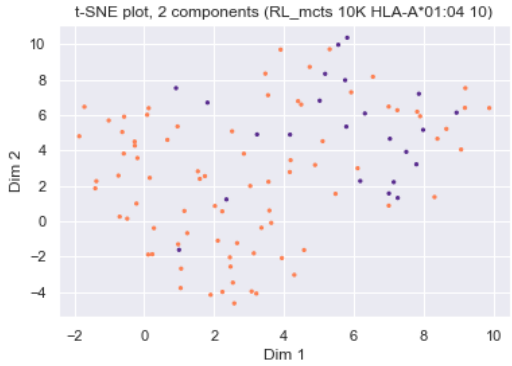




- Allele: HLA-A*01:04 ; Length = 9 & 10

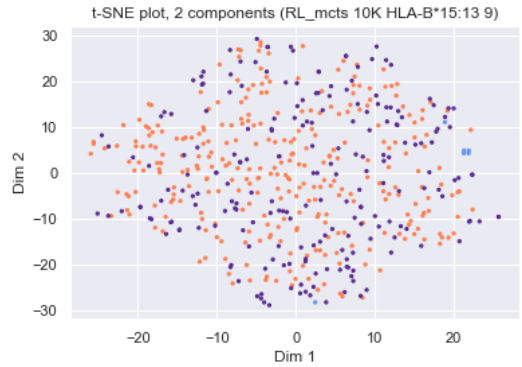
RL 1K (Orange) vs MCTS 10K (Purple)

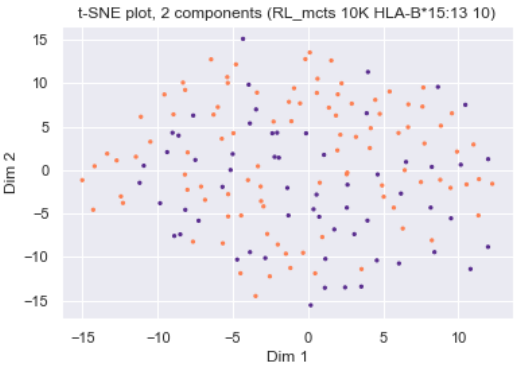




- Allele: HLA-B*15:13 ; Length = 9 & 10

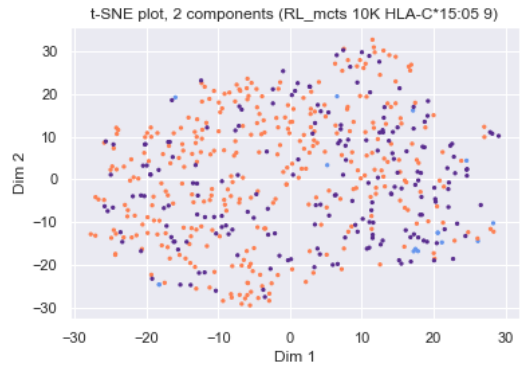
RL 1K vs MCTS 10K

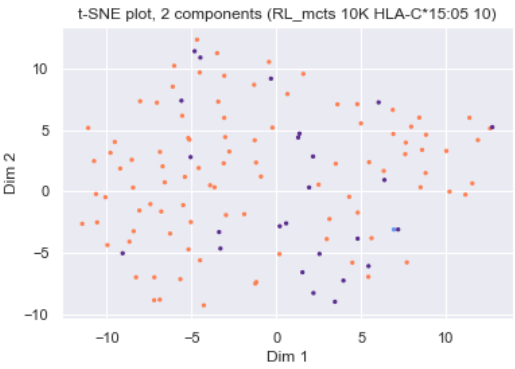




- Allele: HLA-C*15:05 ; Length = 9 & 10

RL 1K vs MCTS 10K





2. Preliminary results of bayesian optimization

The current reconstruct accuracy is about 95%.

The current results of Bayesian optimization with 100 initial points and 1000 iterations are not good.

Possible ways to optimize the results:

1. improve the reconstruct accuracy to 99%
2. try to tune the parameter κ in UCB formula ($UCB(x) = \mu(x) + \kappa * \sigma(x)$)
3. improve the number of iterations or initial points so that the GP within BO can converge.

Note that this BO is an allele-specific model. Remember that GP in BO need to model the predicted presentation score $f(s|a,p)$ between allele and peptide. If we can only maximize peptide sequences, then the input of GP should be only peptide sequence. Therefore, GP actually models the correlation between peptide sequence p and its presentation score with a given allele a, that is $p(s|a,p) = f^a(s|p)$. Because a is not an input of GP model

- Generate 500 peptides for each HLA-A allele in the table using RL model;
- Extract 500 peptides (including positive peptides and negative peptides) with binding affinities for the largest allele in IEDB dataset.
- Build a google site

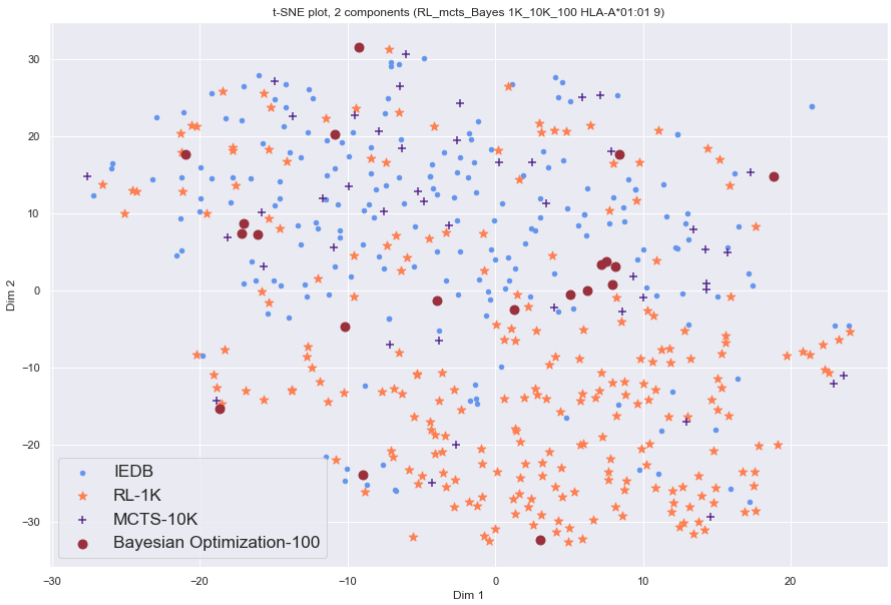
Update (7-7)

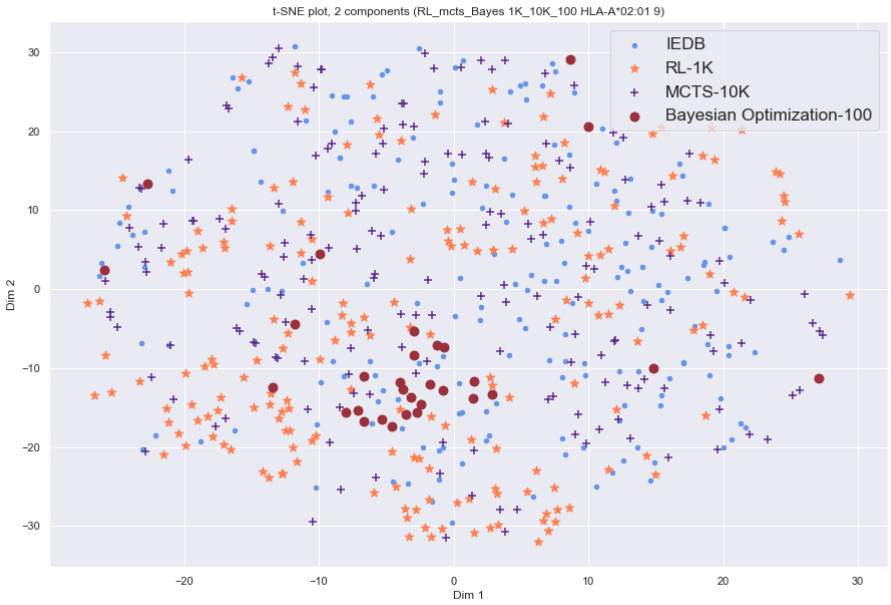
1. Result of Bayesian Optimization

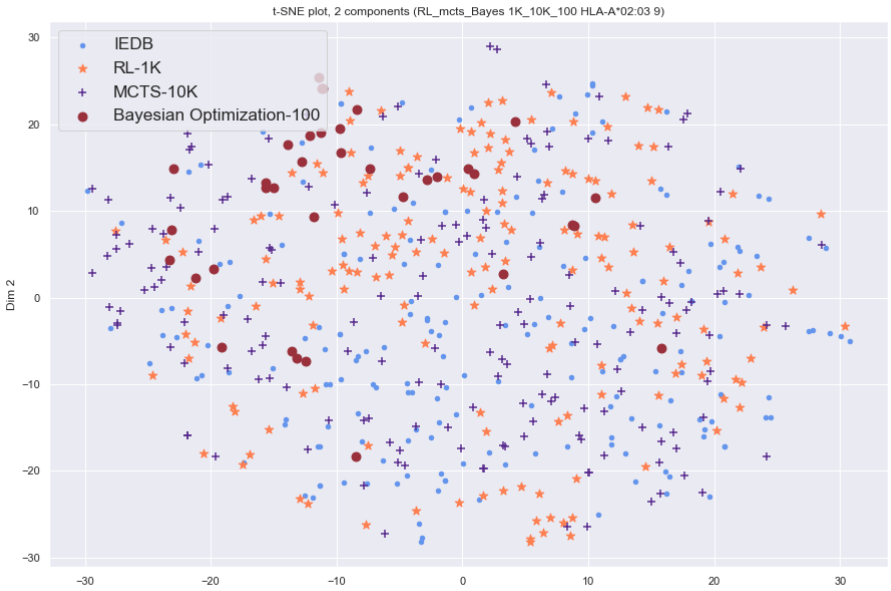
- The previous way I used to run the Bayesian Optimization is incorrect... I ran BO for one time and each BO would optimize the peptides for 1000 steps. However, by checking other papers using the BO, I realized that I should run BO for multiple times and each BO should optimize the peptides for only 5 - 10 steps. In each run, the gaussian process will have different random initial points, and thus the BO could have different optimized peptides.
- This method is very time-consuming. The algorithm has been ran in a paralleled way on 15 cores and still can cost more than 5 hours with 500 initial points (used to train a Gaussian Process model) and only 100 rollouts (i.e., times).
- The number of initial points can significantly influence the running time of this algorithm. The model with 500 initial points achieves significantly better results than that with 100 initial points. However, this model with 500 initial points cost 8 hours to finish the optimization on 23 alleles with 100 rollouts while the model with 100 initial points cost only 2.4 hours.

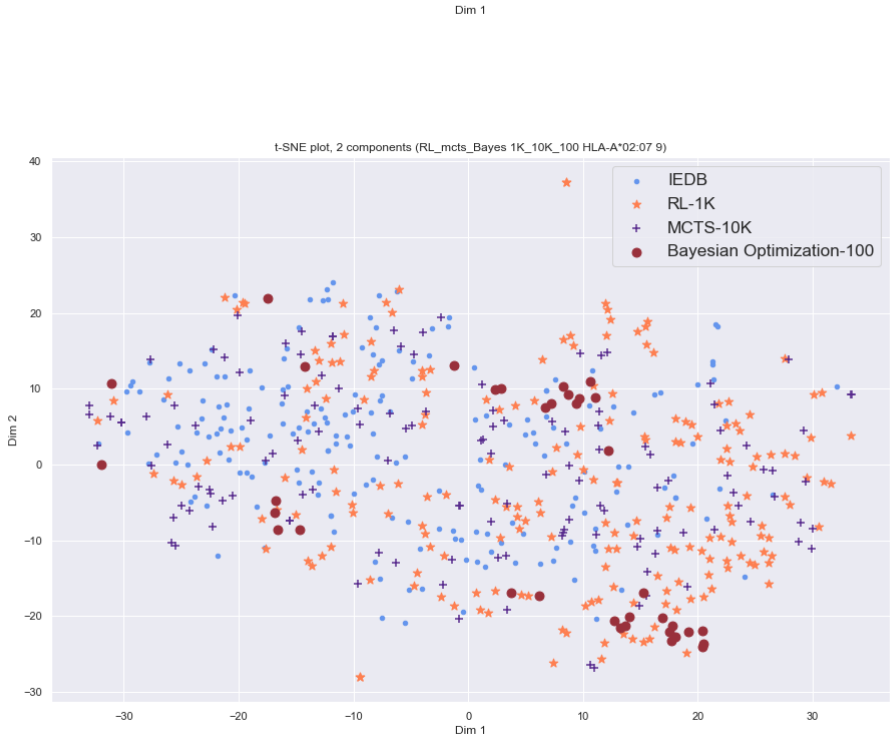
I just found another BO package and will test whether that one is faster than the one I used.

Visualization:


t-SNE plot, 2 components (RL_mcts_Bayes 1K_10K_100 HLA-A*01:01 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
Dim 1
Dim 2


t-SNE plot, 2 components (RL_mcts_Bayes 1K_10K_100 HLA-A*02:01 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
Dim 1
Dim 2


t-SNE plot, 2 components (RL_mcts_Bayes 1K_10K_100 HLA-A*02:03 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
Dim 2



Update (7-12)

1. Result of backpropagation
    - initialize the peptide latent embedding by encoding a random peptide.
    - update the peptide latent embedding with the gradients for 50 iterations.
    - output all the latent embeddings with the predicted scores greater than 0.75.
    - decode these latent embeddings to real peptides.
    - check the presentation scores of these peptides.

Observation:

(1) For different alleles, the backpropagation method can find very different number of unique peptides.

This could be due to that some random peptides are optimized to the same good peptide.

That means Backpropagation cannot find diverse peptides for some alleles.

(2) This method is very efficient. It took about 1.5 hours to get the results with 10K rollouts.

2. Update on Bayesian Optimization

    The package BoTorch is faster than the previous package I used. Now, it took about 70-80 mins to get the results with 100 rollouts. With the previous package, BO with the same parameters can run more than 8 hours.

    With BoTorch, we can save the trained GP model and restore it when training on the new sampled data. The results seem to be better.

    It can cost more than 13 hours to get the results with 1000 rollouts.

    Observation:

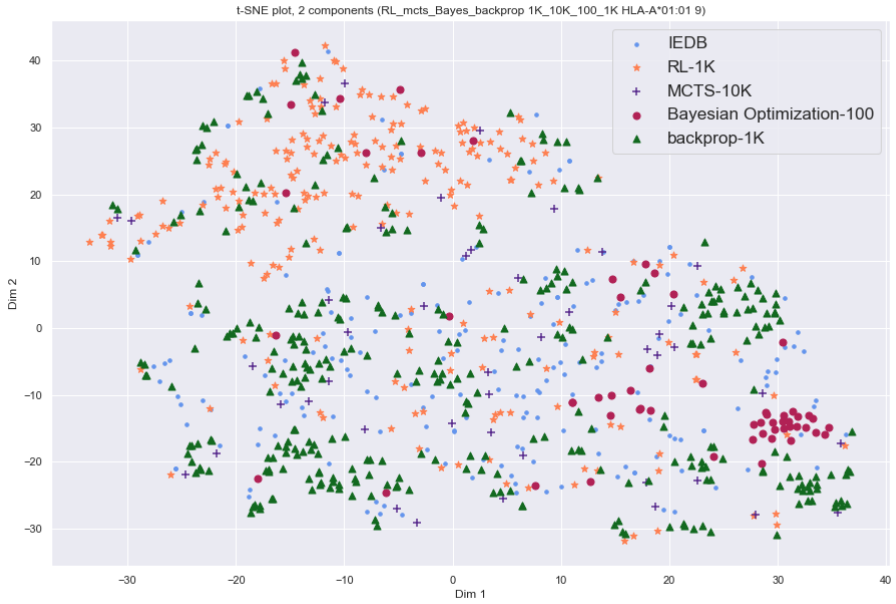
(1) Same with backpropagation, this method can find very different number of unique peptides.

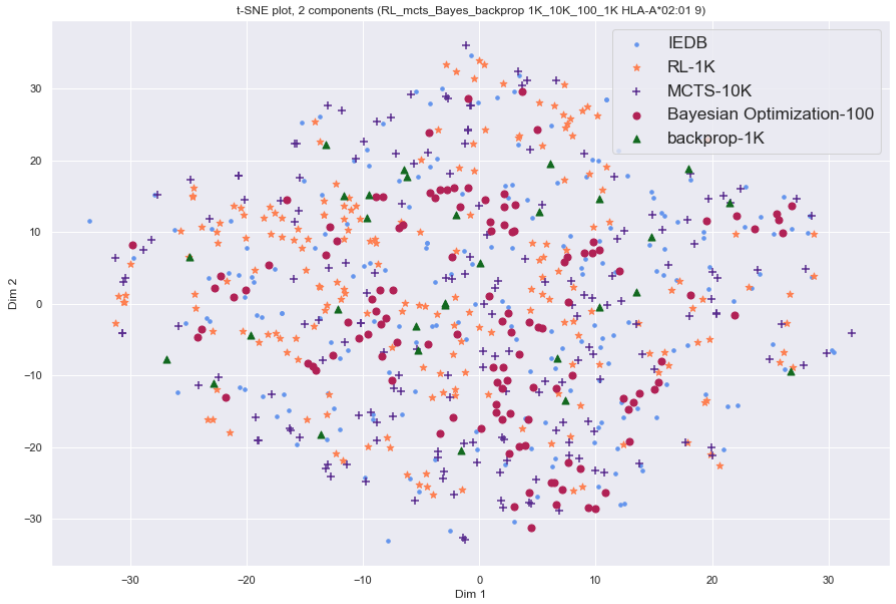
Verification:

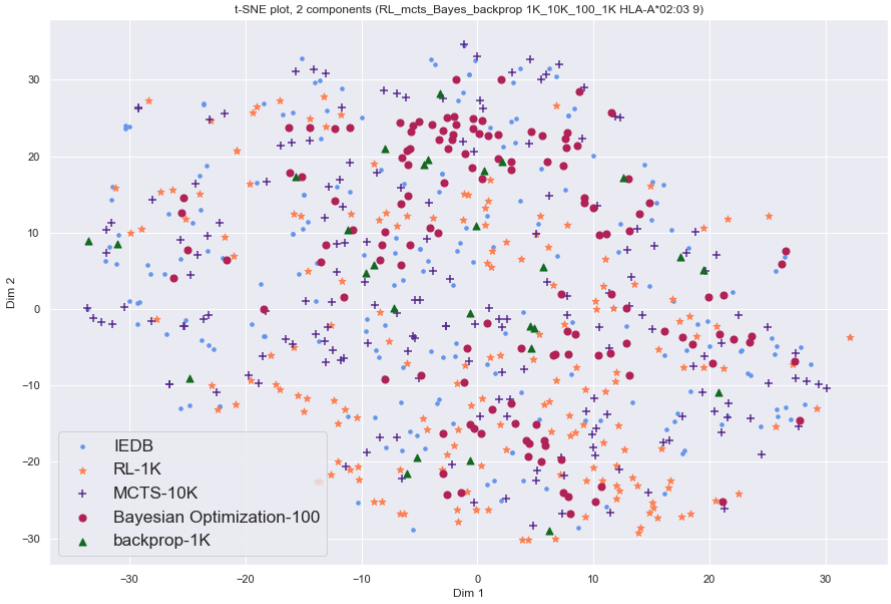
1) Check whether the peptides generated by Backpropagation and BO are similar or not.

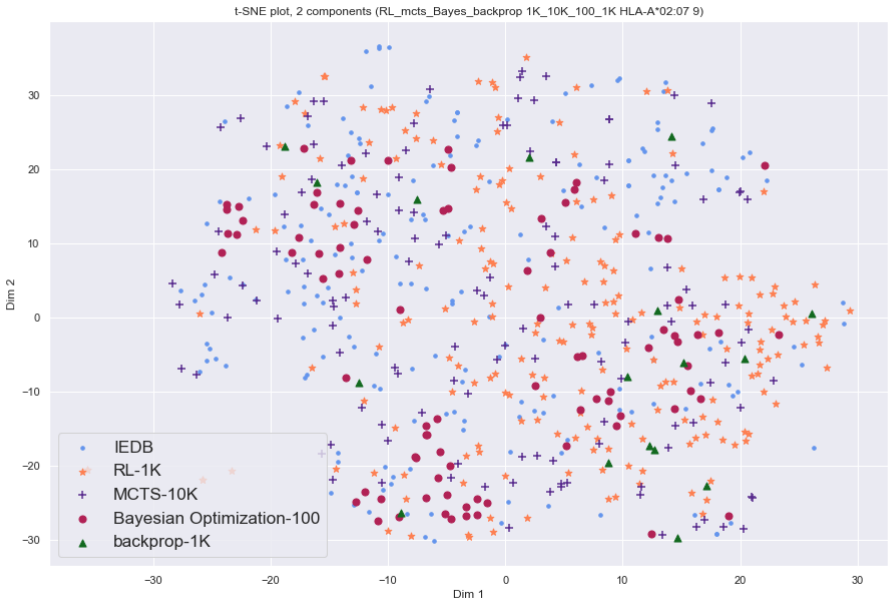
2) Check whether the VAE performs bad on reconstruction of positive peptides of HLA-B5401 and HLA-B5101.

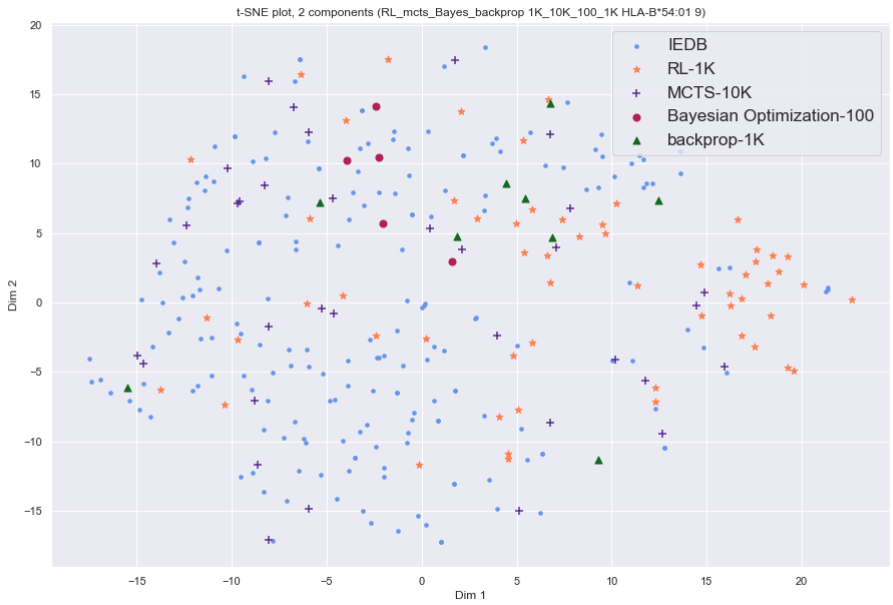
Visualization:

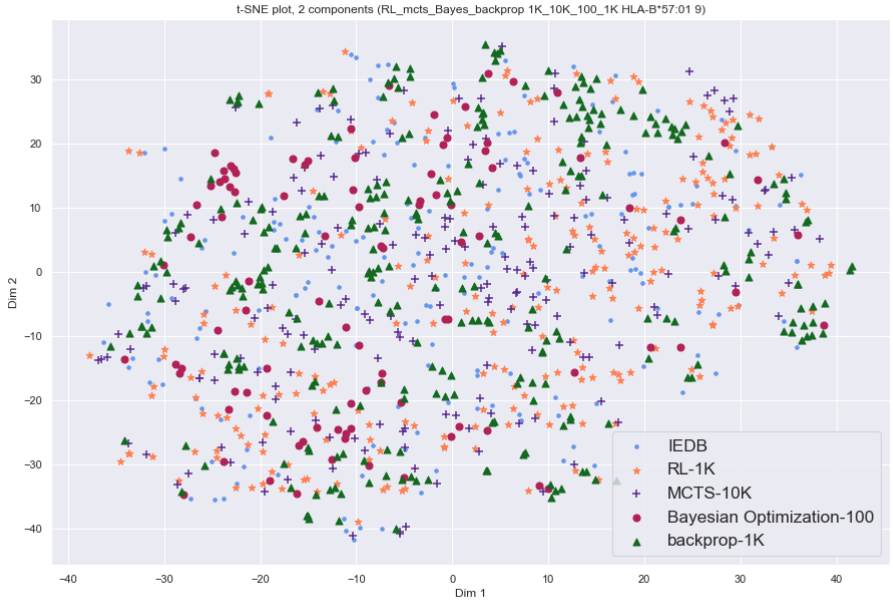


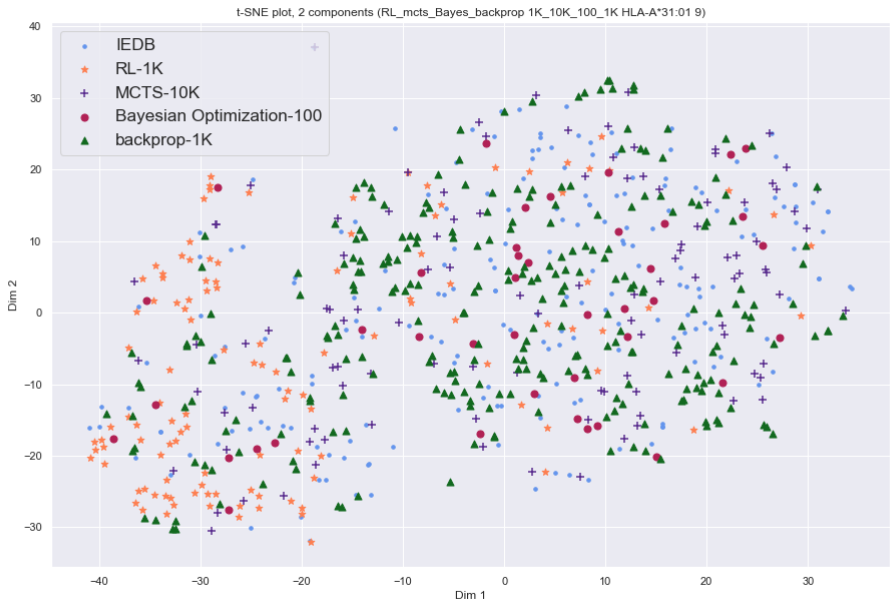



t-SNE plot, 2 components (RL_mcts_Bayes_backprop 1K_10K_100_1K HLA-A*02:03 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
backprop-1K
Dim 1
Dim 2


t-SNE plot, 2 components (RL_mcts_Bayes_backprop 1K_10K_100_1K HLA-A*02:07 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
backprop-1K
Dim 1
Dim 2


t-SNE plot, 2 components (RL_mcts_Bayes_backprop 1K_10K_100_1K HLA-B*54:01 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
backprop-1K
Dim 1
Dim 2


t-SNE plot, 2 components (RL_mcts_Bayes_backprop 1K_10K_100_1K HLA-B*57:01 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
backprop-1K
Dim 1
Dim 2



Metrics:

Quanti metrics

- Diversity/Standard Deviation (of top 100?)
- Memory (leave running time to the)
- count of qualified peptides
- average presentation scores

Current metric - pairwise hamming distance doesn't make sense.

Update (7-23)

1. Random Methods

   With the batch size 128, this random method can generate 300 - 700 peptides per second.

   Among the generated peptides, 0.07% ~ 0.9% peptides are qualified. With 0.07%, this method can generate 1 qualified peptides with 2.1 s ~ 4.8 s. That means, even with the smallest positive rate, this random method is faster than all the other methods.

2. Motif Methods

   For alleles with the data, this motif-based method can generate many positive peptides.

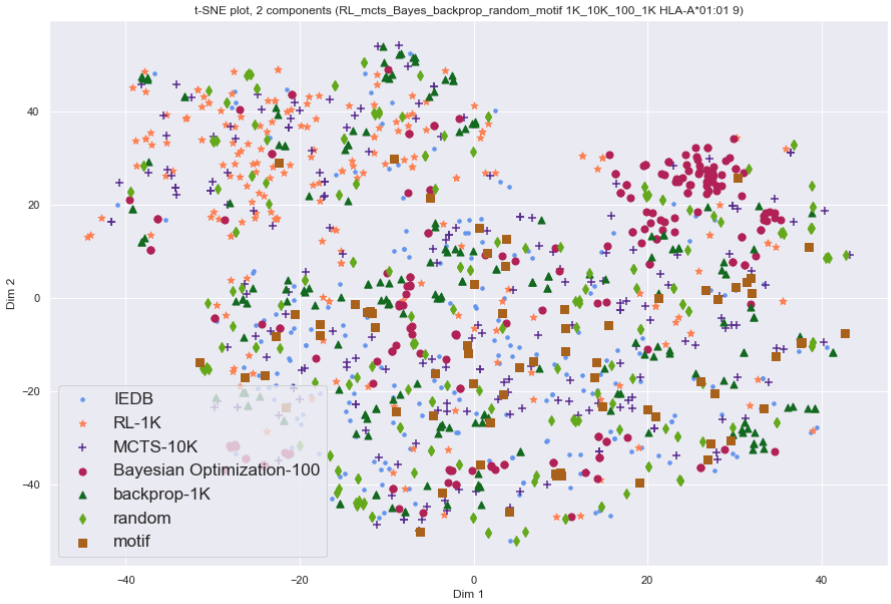
   For alleles without the data, sometimes it can generate qualified peptides as many as the random methods, but sometimes it can not. It depends on the quality of the motif from other alleles.

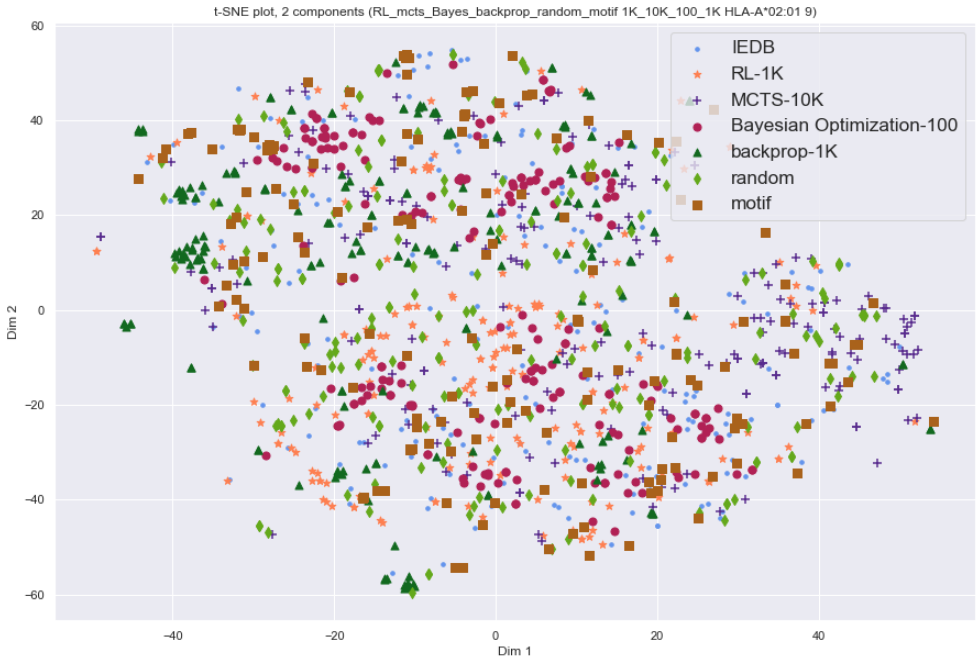
In a conclusion, compared with other baselines and the RL method, random generation and motif-based generation are the most efficient way to do free generation. The high efficiency of these two methods come from the oracle (MHCflurry). Among other baseline methods, at each step, the oracle will be called to evaluate only one peptide, while these two methods asked the oracle to predict 128 peptides.

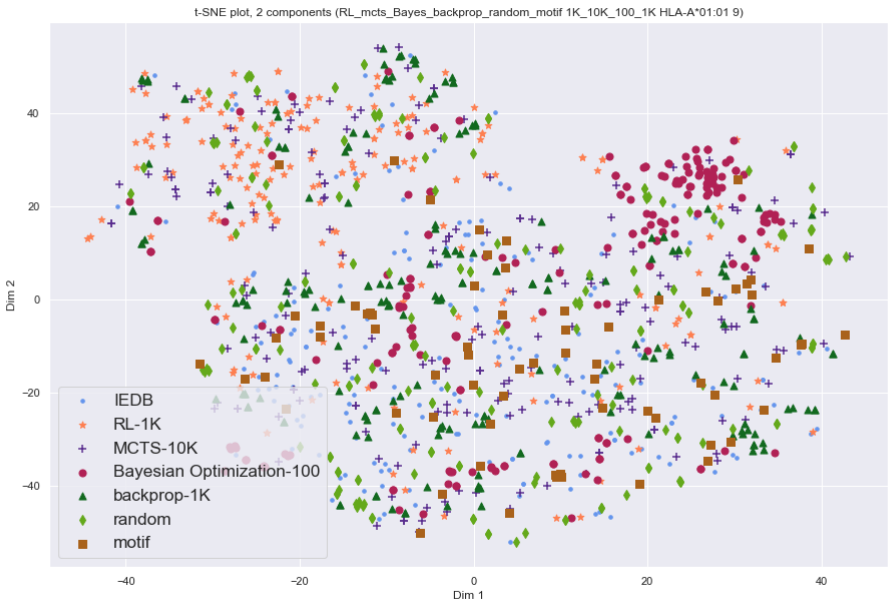
Conclusion: If the objective is to generate sequences without any constraints, and the oracle itself is efficient enough, and the positive rate of random data is not like 0.00[....]01%, using brutal force is a good choice.

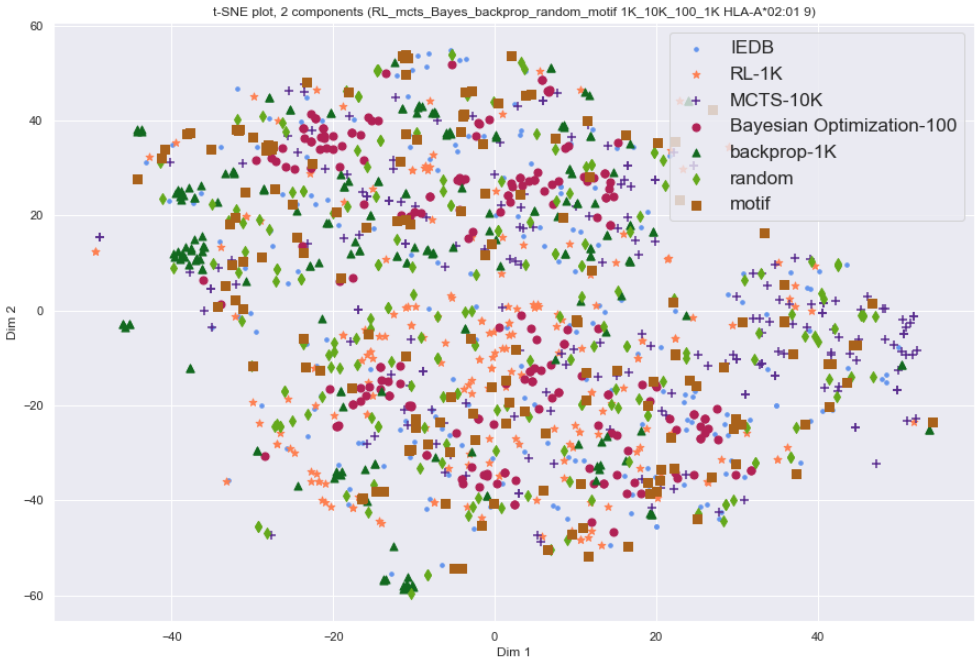
Visualization:

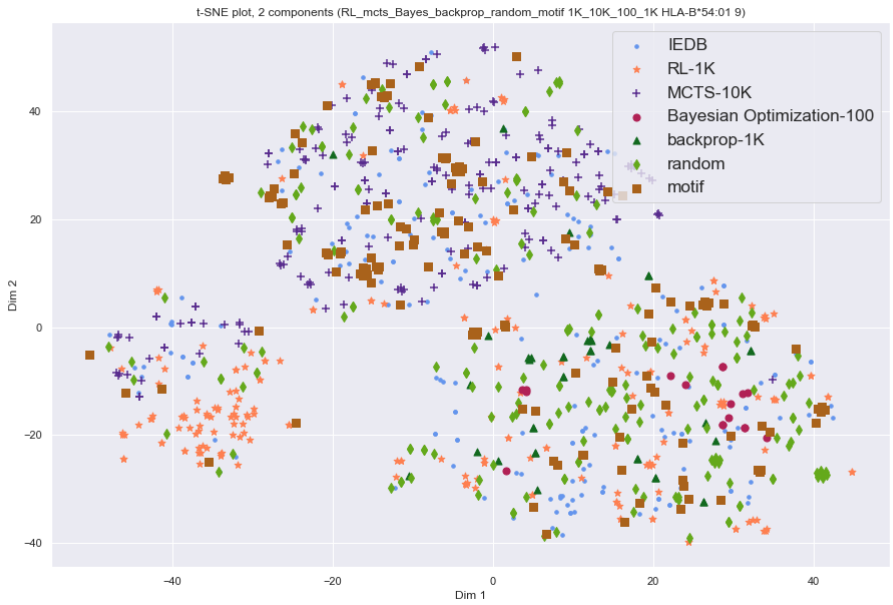
- Alleles with the Data



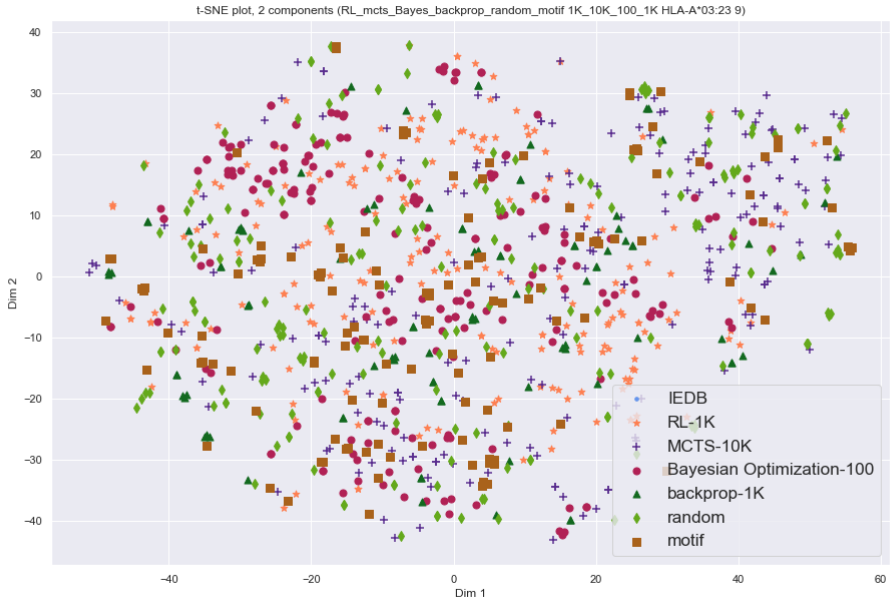


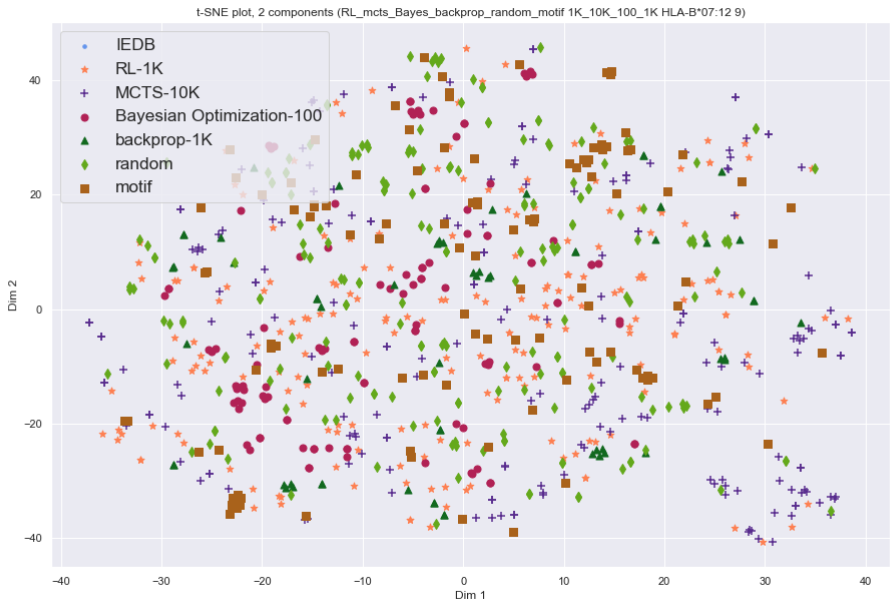

t-SNE plot, 2 components (RL_mcts_Bayes_backprop_random_motif 1K_10K_100_1K HLA-A*01:01 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
backprop-1K
random
motif
Dim 1
Dim 2

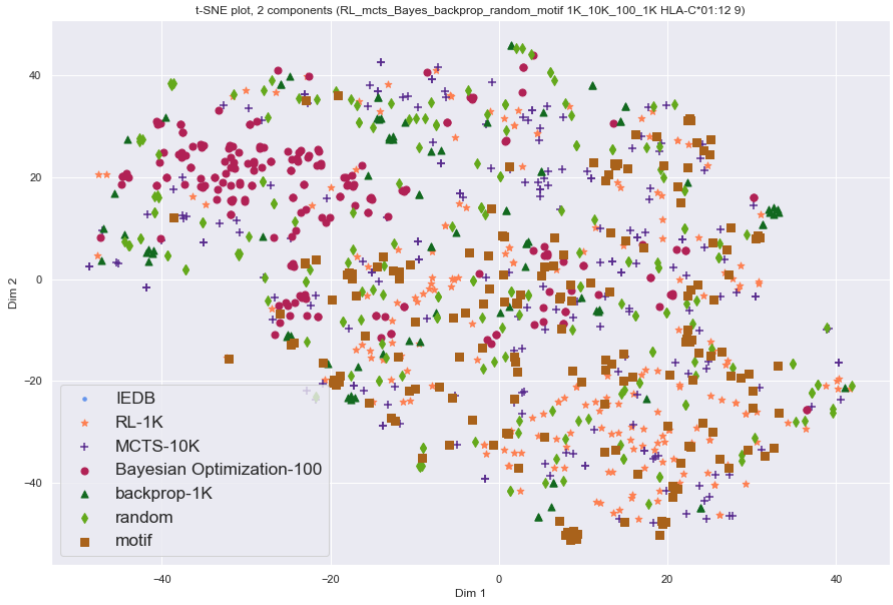

t-SNE plot, 2 components (RL_mcts_Bayes_backprop_random_motif 1K_10K_100_1K HLA-A*02:01 9)
IEDB
RL-1K
MCTS-10K
Bayesian Optimization-100
backprop-1K
random
motif
Dim 1
Dim 2



- Alleles without the Data







How can we demonstrate the performance?

1. We can let all the models to generate the same number of peptides, and count the percentage of positive peptides.

The data of MHC class 2 is more complicated since the length of peptides ranges from 12 to 19.

Update (7-26)

Objective: use **physical model** as reward function to generate peptides for alleles **without data**.

- Why "random" cannot be the solution for the project with physical model ?

  Since we need to use physical model, sample efficiency could be a problem. In this case, we should not use random to generate millions of peptides and evaluate them to get hundreds of positive peptides, because this approach assumes that our oracle (i.e., MHCflurry or Physical model) is very efficient so that millions of peptides can be evaluated.

- Why do we need the RL? Can we directly use "random" or any other baselines on the updated classifier with the augmented data from physical model?

  If we directly use the physical model to generate a batch of pMHC data and then update the classifier, we do not need RL to optimize on that classifier.

  However, by using the physical model in an interactive way, RL could have advantages below:

  (1) achieve better exploration in an efficient way;

  (2) infer motif from the learned policy;

  (3) learn the peptide generation problem with physical model in an active learning way.

  (4) directly learn the signals from physical model.

- How can we solve this "sample efficiency" problem?

  Though we can still use the data from physical model to update MHCflurry and get more accurate predictions, generating data for 10k alleles using physical model is unpractical.

  Therefore, if we want to use physical model, no matter with RL or other baselines (random / backpropagation), we have to make sure that the testing pairs for physical model are very important and can help reduce **uncertainty**.

- How can we quantify the uncertainty?

  The uncertainty of this peptide generation problem comes from two parts: (1) alleles without data; (2) unvisited peptides.

  The most simplest way is to use the uncertainty from the ensemble of networks. But the uncertainty estimation could be improved, for example, we can formulate this problem in an hierarchical way.

  (1) solution for alleles without data;

  Inspired by the uncertainty estimation method for active learning [3], I think maybe we can maintain a graph with all the alleles.

  Each allele represents a node in the graph and has the number of peptides as an attribute. Similar alleles will be connected with the edges. The weight of the edges could be determined by the number of common positive peptides or the similarity on latent embeddings.

  (For RL, the alleles with high uncertainty could be selected with higher probability during the training.)

  (2) solution for unvisited peptides; I don't have a specific idea.....

- How can we implement the uncertainty with RL? any related work?

  I just found a paper that could be related to our objective. The paper name is "MUSBO: Model-based Uncertainty Regularized and Sample Efficient Batch Optimization for Deployment Constrained Reinforcement Learning". They used an ensemble of networks to calculate the uncertainty of each state-action pair. The state-action pair with higher uncertainty will contribute less to the update of agent.

  We can use similar framework to calculate the uncertainty of the predicted reward. The state-action pair with higher uncertainty will be evaluated using physical model.

- If we do not have the physical model available, how can we prepare for the AAAI submission?

  (1) change the current model to model-based. use an ensemble of reward function models to calculate the uncertainty.

  (2) select those pairs with the highest uncertainty for the evaluation of MHCflurry. To make sure the simulation is accurate, we can only use the alleles with the data for training and testing.

  (3) limit the calls of MHCflurry for all the baselines (e.g., 10k calls)? Some baselines are allele-specific..... Maybe we can use only pan-allele methods such as random methods and backpropagation methods.

  We can even conduct an ablation study to evaluate whether the performance of backpropagation method can increase with the selected uncertain pMHC data from RL.

---

1. Pardo, Fabio, et al. "Time limits in reinforcement learning." *International Conference on Machine Learning*. PMLR, 2018. ↩
2. Killoran, Nathan, et al. "Generating and designing DNA with deep generative models." *arXiv preprint arXiv:1712.06148* (2017). ↩
3. Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization ↩