Supporting Information for “ Optimal Learning for Discovering and Optimizing Enzymatic Peptide Substrates ”

November 30, 2016

Contents

SI Text

1 Peptide Optimization with Optimal Learning

1.1 Problem Formulation

We first introduce mathematical notation that allows precise discussion of the POOL methodology. Let E be the set of peptides over which we would like to search. In our application, the set E will be the set of peptides of length less than 38 amino acids. For any peptide $e \in E$, let $y(e) \in \{0, 1\}$ be a binary label indicating whether the peptide e is a "hit". We suppose that this label can be observed directly through experiment, but is otherwise unknown *a priori*. In our application, we consider two definitions of $y(e)$, depending on whether we are searching for peptides with Sfp-type or AcpS-type activity:

- When searching for peptides with Sfp-type activity, we let $y(e)$ be 1 if the peptide is a substrate for both Sfp and AcpH, but is not a substrate for AcpS. We call this an "Sfp-type hit".
- When searching for peptides with AcpS-type activity, we let $y(e)$ be 1 if the peptide is a substrate for both AcpS and AcpH, but is not a substrate for Sfp. We call this an "AcpS-type hit".

POOL searches for peptides that are hits, and for which another given fitness function $f(e)$ is large. We assume that this fitness function can be observed directly without requiring a physical experiment. In our application, the fitness is negative one times the length of the peptide. The negative one is present because we will want fitness to be large, but we want the length to be small.

Our goal, in terms of the notation we have defined, is to find a peptide $e \in E$ for which $y(e) = 1$ and for which $f(e)$ is as large as possible. We can write this problem as,

$$\arg\max_{e \in E, y(e)=1} f(e). \tag{1}$$

However, we typically cannot solve this problem directly because $y(e)$ has only been observed for those peptides on which we have performed an experiment, which is typically only a small fraction of the peptides in our search space E, and the additional experimental effort required to observe $y(e)$ for all peptides in E is simply beyond the realm of feasibility. For example, in our application, E contains more than 2×10^{49} peptides. Our current experimental approach allows testing roughly 600 peptides for one round, and we can only perform 4 rounds of experiments in total. Therefore, we only have the budget to test a tiny fraction of E in searching for peptide hits.

POOL is designed for these situations in which the number of peptides we can evaluate is substantially smaller than the search space. It assumes that we have available the values of $y(e)$ for a small number of previously evaluated peptides (we refer to this as "training data"), and that we can also evaluate $y(e)$ for a set of new peptides chosen by POOL. POOL supposes that peptides are evaluated in batches, and we let M denote the number of peptides in a batch. POOL may be applied once, to recommend a single batch of peptides to test, or it may be applied repeatedly, each time adding the results from previously tested batches to the training data.

POOL uses such limited evaluation budgets to best support solving (1), in the specific sense of maximizing the probability of finding a peptide hit whose quality is better than some target quality b. It consists of two steps: first, a prediction step, in which a machine learning method (Bayesian Naïve Bayes classifier, customized for peptide activity) provides a joint probability distribution over peptide activity; second, a probability-of-improvement step, in which we use value-of-information analysis to define the quality of a set of peptides to test, and then use an approximation algorithm to find a set of peptides to test that provides a large probability-of-improvement.

We discuss POOL in more detail in the following sections: Sect. 1.2 describes the Bayesian Naïve Bayes classifier that predicts the peptide activity; Sect. 1.3 provides the value-of-information analysis and the approximation algorithm in POOL; Sect. 1.4 discusses why POOL performs better than the "predict-then-optimize" approach.

1.2 Prediction Model: Bayesian Naïve Bayes Classifier

In the prediction step of POOL, we build a Bayesian Naïve Bayes classifier as the underlying machine learning model to predict the joint probability distribution over peptide activity.

Naïve Bayes classifiers [3, 4] are popular and extensively studied in text classification problems because of their fast computation and simple implementation. Peptides are sequences of amino acids, which have big influence on peptide activity, and this is very similar to text classification, where words within text influence on which class the text belongs to, therefore, a Naïve Bayes classifier is suitable for peptide activity prediction. In addition, we usually do not afford to have a training dataset with thousands of data points in peptide activity, the independence assumption across features in Naïve Bayes allows it to achieve good prediction performance even when the data size is small, which is desirable in the problems that we are dealing with.

To describe the model, let $X = (X_1, \ldots, X_L)$ be a feature vector and Y be its label. Using Bayes' Rule, we have:

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(X = x)} = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\sum_{y'} \mathbb{P}(X = x|Y = y')\mathbb{P}(Y = y')}.$$

The Naïve Bayes classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable, i.e.

$$\mathbb{P}(Y = y|X = x) = \frac{\prod_{j=1}^{L} \mathbb{P}(X_j = x_j|Y = y)\mathbb{P}(Y = y)}{\sum_{y'} \prod_{j=1}^{L} \mathbb{P}(X_j = x_j|Y = y)\mathbb{P}(Y = y)}.$$

In the context of our application, Y is a random variable that is either "1" or "0", depicts whether a peptide is a "hit" or "miss"; X is feature vector that encodes any peptide of interest, which is designed in the following: since a peptide is a sequence of amino acids, we use $A_j \in \{1, \ldots, 20\}$ to indicate the type of amino acid in jth position of the sequence, where $j = 1, \ldots, L$; we further partition the 20 different types of amino acids into K groups ($K \leq 20$) according to their biochemical similarities, then jth feature is simply indicating which group the amino acid in the jth position of the sequence belongs to, denoted as x_j.

We let $\theta_{y,j}(k) = \mathbb{P}(X_i = k|Y(X) = y)$, where $j = 1, \ldots, L$, $k = 1, \ldots, K$ and $y \in \{0, 1\}$. Then given a prior distribution $\mathbb{P}(Y(x) = y)$, $y \in \{0, 1\}$, the Bayesian inference for any peptide x is

$$\mathbb{P}\left(Y(x) = 1|\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, x\right) = \frac{\mathbb{P}(Y(x) = 1) \prod_j \theta_{1,j}(x_j)}{\left[\mathbb{P}(Y(x) = 1) \prod_j \theta_{1,j}(x_j)\right] + \left[\mathbb{P}(Y(x) = 0) \prod_j \theta_{0,j}(x_j)\right]}, \tag{2}$$

where $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ are the matrix representations of $\theta_{0,j}(k)$ and $\theta_{1,j}(k)$.

We train the model parameters $\theta_{y,j}(k)$ using Bayesian inference. We assume the prior distribution over $\theta_{y,j}(\cdot)$ is Dirichlet$(\alpha_{y,j}(1), \ldots, \alpha_{y,j}(K))$, and we can encode prior knowledge from domain experts into the choices of the prior parameters $\alpha_{y,j}(k)$. In this biochemical application, the enzymes can only attach/detach "label" to/from a special position in the peptide, and we know that those positions further away from this special position will have less influence on whether the peptide is a hit or not. To encode this information into the prior, we increase the value of $\alpha_{y,j}(k)$ and make it uniform over k as j moves further away from the known position, and this encodes the belief that the positions far away from the special position is less sensitive about which amino acid to appear over that position. From Bayesian statistics, we know that the posterior is $\text{Dirichlet}(\alpha_{y,j}(1) + N_{y,j}(1), \ldots, \alpha_{y,j}(K) + N_{y,j}(K))$, where $N_{y,j}(k)$ is the number of the peptides in the training set with $Y(x) = y$, and $x_j = k$. Now we have constructed the model for predicting peptide activity and the method of updating model parameters given data.

1.3 The Contingency Plan: Value-of-information Analysis

We have described how POOL method builds a contingency plan in finding the peptides that are likely to be hits in the main article. In this section, we show how this approach is derived mathematically. We provide an outline for this section here: Sect. 1.3.1 discusses value-of-information analysis and gives the high-level strategy of constructing the set of peptides to test next; Sect. 1.3.2 provides an approximation algorithm to solve the otherwise difficult-to-solve objective given in 1.3.2, and proves that this approximation algorithm has a performance guarantee; Sect. 1.3.3 further simplifies the approximation algorithm, and derives the main procedures of POOL; Sect. 1.3.4 and 1.3.5 provide two implementations of the approximation algorithm.

1.3.1 Probability-of-improvement

Given the prediction model, the Bayesian optimization approach uses decision theory to suggest which points in the function's domain would be the most valuable to evaluate next, known as the value of information analysis. In our application, a point in the function's domain corresponds to a peptide, and therefore, we use value-of-information analysis to decide the set of peptides to test next in finding peptide hits. Specifically, we use an information criterion, which is a function, to quantify the information gain if we were to test a given set of peptides, and then our goal is to find the set of peptides that maximizes this information gain.

Probability-of-improvement is one such criterion, initially proposed by [2]. In this context, the probability-of-improvement quantifies the probability that a set of peptides S, if tested, will reveal a hit whose fitness $f(e)$ improves on some benchmark value b, which is typically the best $f(e)$ of hits observed in the past. We define this probability-of-improvement, $\text{PI}(S)$, to be

$$\text{PI}(S) = \mathbb{P}\left(\max_{e \in S, y(e)=1} f(e) > b\right). \tag{3}$$

To ensure the event that none of the peptides we test is hit does not contribute to probability-of-improvement, we define max over an empty set is $-\infty$. We find S by solving

$$\underset{S \subseteq E, |S| \leq M}{\arg\max}\ \text{PI}(S), \tag{4}$$

where M is the number of peptides that can be tested simultaneously in a single round of experiment.

After we find the set S and test this set of peptides, we obtain the labels for these peptides that whether they are hits or not. We update our prediction model in Sect. 1.2 with the new data, and solve (4) again to obtain a new set of peptides to test for the next round of experiment. We repeat this process until we find enough hits or resource is exhausted. We have just described the high-level approach of POOL, however, solving (4) when E and M are large is very difficult. In this application, the size of E is in the order of 10^{49} and $M \approx 600$, and therefore brute-force approach is not feasible. The rest of the section is dedicated to proposing an approximation algorithm to solve (4), and proving its performance guarantee.

1.3.2 Greedy Algorithm and Its Performance Guarantee

Directly solving (4) is computationally challenging, because the size of the search space $|E|^M$, which grows exponentially in M. Therefore we propose to use greedy algorithm to solve it approximately, that is, we begin with $S = \emptyset$, and add one peptide at a time to S by solving

$$\operatorname*{arg\,max}_{e \in E \setminus S} \mathrm{PI}(S \cup \{e\}), \tag{5}$$

until $|S| = M$. This algorithm is much less computational challenge because we reduced the size of the search space from $|E|^M$ down to $|E|$. We also show in the following theorem that the this algorithm has a performance guarantee compared to the optimal solution.

Theorem 1. The greedy algorithm always produces a solution whose probability-of-improvement is at least $1 - [(k-1)/k]^k \geq 1 - 1/e$ times the optimal solution of (4).

To prove the theorem, we use the following two lemmas:

Lemma 1. If $F(S)$ is submodular, non-decreasing and $F(\emptyset) = 0$, the greedy heuristic always produces a solution whose value is at least $1 - [(k-1)/k]^k$ times the optimal value, where $|S| \leq k$. This bound can be achieved for each k and has a limiting value of $1 - 1/e$, where e is the base of the natural logarithm.

Lemma 1 is a result from the analysis of greedy heuristics in combinatorial optimization by Nemhauser [5], which provides lower bound on the greedy algorithm given that the objective function satisfies certain conditions. To show that this lemma applies to Theorem 1, we need to show our objective function, which is probability-of-improvement, satisfies the conditions given in Lemma 1. Lemma 2 to be stated below shows that probability-of-improvement indeed satisfies these conditions, and therefore concludes the theorem.

Lemma 2. Probability-of-improvement $\mathrm{PI}(S)$ is submodular, non-decreasing and $\mathrm{PI}(\emptyset) = 0$.

Proof. Let $f^*(S) = \max_{e \in S, y(e)=1} f(e)$. First we show $\mathrm{PI}(\emptyset) = 0$.

$$\mathrm{PI}(\emptyset) = \mathbb{P}(f^*(\emptyset) > b) = \mathbb{P}(-\infty > b) = 0.$$

To show $\mathrm{PI}(S)$ is non-decreasing, let $A \subseteq B \subseteq E$ where E is a finite set, then

$$\begin{aligned}
\mathrm{PI}(B) &= \mathbb{P}(f^*(B) > b) \\
&= \mathbb{P}(f^*(B) > b \mid f^*(A) \leq b)\mathbb{P}(f^*(A) \leq b) + \mathbb{P}(f^*(B) > b \mid f^*(A) > b)\mathbb{P}(f^*(A) > b) \\
&= \mathbb{P}(f^*(B) > b \mid f^*(A) \leq b)\mathbb{P}(f^*(A) \leq b) + \mathbb{P}(f^*(A) > b) \\
&\geq \mathbb{P}(f^*(A) > b) \\
&= \mathrm{PI}(A).
\end{aligned}$$

Lastly, we want to show $\mathrm{PI}(S)$ is submodular. For $e \in E \backslash B$,

$$\begin{aligned}
&\mathrm{PI}(A \cup \{e\}) - \mathrm{PI}(A) \\
&= \mathbb{P}(f^*(A \cup \{e\}) > b) - \mathbb{P}(f^*(A) > b) \\
&= \mathbb{P}(f^*(A \cup \{e\}) > b \mid f^*(A) > b)\mathbb{P}(f^*(A) > b) \\
&\quad + \mathbb{P}(f^*(A \cup \{e\}) > b \mid f^*(A) \leq b)\mathbb{P}(f^*(A) \leq b) - \mathbb{P}(f^*(A) > b) \\
&= \mathbb{P}(f^*(A \cup \{e\}) > b \mid f^*(A) \leq b)\mathbb{P}(f^*(A) \leq b) \\
&= \mathbb{P}(f(e) > b, y(e) = 1 \mid f^*(A) \leq b)\mathbb{P}(f^*(A) \leq b) \\
&= \mathbb{P}(f(e) > b, y(e) = 1, f^*(A) \leq b).
\end{aligned}$$

Using similar argument,

$$\begin{aligned}
&\mathrm{PI}(B \cup \{e\}) - \mathrm{PI}(B) \\
&= \mathbb{P}(f(e) > b, y(e) = 1, f^*(B) \leq b) \\
&= \mathbb{P}(f(e) > b, y(e) = 1, f^*(A) \leq b, f^*(B \backslash A) \leq b).
\end{aligned}$$

Therefore, $\mathrm{PI}(A \cup \{e\}) - \mathrm{PI}(A) \geq \mathrm{PI}(B \cup \{e\}) - \mathrm{PI}(B)$, thus $\mathrm{PI}(S)$ is submodular. $\square$

Remark. The theoretical result we have obtained so far does not use any information about the prediction model we used in Sect. 1.2, which means Theorem 1 is a general result for our greedy algorithm, independent from the choice of prediction model. Therefore, even one uses a different prediction model than in Sect. 1.2 in this greedy algorithm, the same result applies.

1.3.3 A Simplified Form of the Greedy Algorithm

(5) is still hard to solve, at least in a naive way, where we simply enumerate all peptides $e \in E \backslash S$ and evaluate $\mathrm{PI}(S \cup \{e\})$ for each of them, and then select the peptide for which the PI is the largest. It is hard because first, the peptide space E in our application is too large to enumerate over all of the peptides, and second, computing $\mathrm{PI}(\cdot)$ for each enumeration is expensive because its computational cost increases linearly in the size of S. We can transform (5) into a simplified form so that 1. we solve the same problem without having to evaluate $\mathrm{PI}(\cdot)$, and 2. it paves the way for using more efficient methods to solve the problem as discussed in Sect. 1.3.4 and 1.3.5. We show the simplified form in the following proposition:

Proposition 1. The solution to (5) is equivalent to

$$\underset{e \in E \backslash S, f(e) > b}{\arg\max} \ \mathbb{P}(y(e) = 1 \mid y(e') = 0, \forall e' \in S). \tag{6}$$

Proof. The objective in (5) can be written as

$$\begin{aligned}
&\mathrm{PI}(S \cup \{e\}) \\
&= \mathbb{P}(f^*(S \cup \{e\}) > b), \\
&= \mathbb{P}(f^*(S) > b) + \mathbb{P}(f^*(S) \leq b)\mathbb{P}(f(e) > b, y(e) = 1 \mid f^*(S) \leq b).
\end{aligned}$$

Because S is a constant in (5), the following equality holds:

$$\underset{e \in E \backslash S}{\arg\max} \, \mathrm{PI}(S \cup \{e\}) = \underset{e \in E \backslash S}{\arg\max} \, \mathbb{P}(f(e) > b, y(e) = 1 \mid f^*(S) \leq b). \tag{7}$$

When $f(e) \leq b$, the objective in (7), $\mathbb{P}(f(e) > b, y(e) = 1 \mid f^*(S) \leq b)$ evaluates to zero; when $f(e) > b$, this objective is always greater or equal to zero because it is a probability function. Therefore, (7) is equivalent to

$$\underset{e \in E \setminus S, f(e) > b}{\arg\max} \; \mathbb{P}(y(e) = 1 \mid y(e') = 0, \forall e' \in S).$$

$\square$

The formulation in Proposition 1 conveys exactly the same message as the verbal description of the POOL method in the main text: we construct the set S by adding the top-scoring peptide iteratively, i.e., the most probable being a hit, according to the prediction model trained on all previously observed data plus the peptides that are already in S and assume none of them is a hit. Here we provided the mathematical derivation for the POOL method.

Note that the derivation so far remains independent from the choice of prediction model, and therefore, the POOL method and its performance guarantee apply to any prediction model as long as the objective in (6) can be evaluated.

1.3.4 POOL-MINLP: Solve the Greedy Step using MINLP

Solving (6) by simply brute-force search still remains infeasible when the search space E is large. In our application, E contains more than 2×10^{49} peptides, and if evaluating for one peptide takes $\sim 10^{-3}$ seconds, it will require more than 2×10^{46} seconds to complete all evaluations.

To solve this problem more efficiently, we formulate (6) as an instance of Mixed-Integer Nonlinear Program (MINLP), and then we can use any MINLP solver to solve (6) efficiently. Let $\theta_{x_j}^{y',j} = \mathbb{P}(x_j \mid y(e) = y')$, from (2) we have

$$\begin{aligned} \mathbb{P}(y(e) = 1 \mid \boldsymbol{x}) &= \frac{\prod_{j=1}^{J} \theta_{x_j}^{1,j} \mathbb{P}(y(e) = 1)}{\prod_{j=1}^{J} \theta_{x_j}^{1,j} \mathbb{P}(y(e) = 1) + \prod_{j=1}^{J} \theta_{x_j}^{0,j} \mathbb{P}(y(e) = 0)}, \\ &= \frac{\prod_{j=1}^{J} \eta_{x_j}^{j}}{\prod_{j=1}^{J} \eta_{x_j}^{j} + \frac{\mathbb{P}(y(e)=0)}{\mathbb{P}(y(e)=1)}}, \end{aligned} \tag{8}$$

where $\eta_{x_j}^{j} = \frac{\theta_{x_j}^{1,j}}{\theta_{x_j}^{0,j}}$ for $\forall j \in \{1, \ldots, J\}$. Since $\theta_{x_j}^{y'}$s are unknown model parameters, we use Bayesian inference as described in Sect. 1.2 to estimate the posterior probability over them. We then compute expectation of (8) using the posterior probability distribution, written as

$$\hat{\mathbb{P}}(y(e) = 1 \mid \boldsymbol{x}) = \mathbb{E}\left[\frac{\prod_{j=1}^{J} \eta_{x_j}^{j}}{\prod_{j=1}^{J} \eta_{x_j}^{j} + \frac{\mathbb{P}(y(e)=0)}{\mathbb{P}(y(e)=1)}}\right]. \tag{9}$$

To compute (9), we use sample average approximation [1], in which we sample R i.i.d. samples from the posterior distributions of $\boldsymbol{\theta}^{1,j}$ and $\boldsymbol{\theta}^{0,j}$ (they are Dirichlet distributions with parameters given in Sect. 1.2). We then approximate (9) by

$$\hat{\mathbb{P}}(y(e) = 1 \mid \boldsymbol{x}) = \frac{1}{R} \sum_{r=1}^{R} \frac{\prod_{j=1}^{J} \eta_{x_j}^{j,r}}{\prod_{j=1}^{J} \eta_{x_j}^{j,r} + \frac{\mathbb{P}(y(e)=0)}{\mathbb{P}(y(e)=1)}}. \tag{10}$$

Now we can replace the objective in (6) by (10), and formulate (6) into a MINLP:

$$\begin{aligned} \max \quad & \sum_{r=1}^{R} \frac{\prod_{j=1}^{J} \sum_{k=1}^{K_j} z_k^j \eta_k^{j,r}}{\prod_{j=1}^{J} \sum_{k=1}^{K_j} z_k^j \eta_k^{j,r} + \frac{\mathbb{P}(y(e)=0)}{\mathbb{P}(y(e)=1)}}, \\ \text{s.t} \quad & z_k^j \in \{0,1\}, \\ & \sum_{k=1}^{K_j} z_k^j = 1, \ \forall j \in \{1,\dots,J\}, \end{aligned} \tag{11}$$

where z_k^j is indicative variables that equal to 1 if $x_j = k$ and 0 otherwise, and $\eta_k^{j,r}, r = 1,\dots,R$ are computed using i.i.d. samples from the posterior distribution of $\boldsymbol{\theta}^{1,j}$ and $\boldsymbol{\theta}^{0,j}$ given the previously observed data plus the additional "data" $\{y(e') = 0, \forall e' \in S\}$. Suppose the solution to (11) is $\boldsymbol{z}^*$, we can translate it to the feature representation $\boldsymbol{x}^*$ by letting $x^{*j} = \sum_k k \cdot z_k^{*j}$.

We summarize the algorithm below, in which the output S is the set of peptides to test next.

Algorithm 1. (Greedy Algorithm for POOL-MINLP)

```
Require: Inputs $M, J, R, K_j, j = 1,\dots,J$, training dataset $\mathcal{D} = \{(\boldsymbol{x}^1, y^1),\dots,(\boldsymbol{x}^N, y^N)\}$ and parameters for the prior distributions $\boldsymbol{\alpha}^{y,j}$.
 1: $S \leftarrow \emptyset$
 2: Calculate posterior distribution of $\boldsymbol{\theta}^{1,j} \sim$ Dirichlet($\boldsymbol{\theta}^{1,j} \mid \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D}, y(\boldsymbol{x}) = 1\}$) for $j = 1,\dots,J$.
 3: for $m = 1$ to $M$ do
 4:    Calculate posterior distribution of $\boldsymbol{\theta}^{0,j} \sim$ Dirichlet($\boldsymbol{\theta}^{0,j} \mid \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D}, y(\boldsymbol{x}) = 1\} \cup S$) for $j = 1,\dots,J$.
 5:    for $r = 1$ to $R$ do
 6:       for $j = 1$ to $J$ do
 7:          Sample $\boldsymbol{\theta}^{1,j}$ from Dirichlet($\boldsymbol{\theta}^{1,j} \mid \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D}, y(\boldsymbol{x}) = 1\}$) and $\boldsymbol{\theta}^{0,j}$ from Dirichlet($\boldsymbol{\theta}^{0,j} \mid \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D}, y(\boldsymbol{x}) = 1\} \cup S$).
 8:          Calculate $\boldsymbol{\eta}_i^{j,r} = \frac{\theta_i^{1,j}}{\theta_i^{0,j}}, \forall i \in K_j$.
 9:       end for
10:    end for
11:    Solve the MINLP in (11) to find $\boldsymbol{z}^*$ and the corresponding peptide $e$.
12:    $S \leftarrow (S, e)$.
13: end for
```

1.3.5 POOL-MAP: Solve the Greedy Step using MAP Estimation

In our application, due to the large peptide space, the MINLP approach still takes hours to solve (6), and since $M = 600$ in the application, the overall time to complete Algorithm 1 is 600 times the a few hours for solving (6), which is too long given the time constraint.

To deal with this problem, we propose a simpler approach using Maximum a Posteriori (MAP) estimation [6]. Although this approach less accurate than the MINLP approach, the computation is much cheaper and faster.

We first obtain the posterior distributions of $\boldsymbol{\theta}^{1,j}$ and $\boldsymbol{\theta}^{0,j}$ for $\forall j \in \{1,\dots,J\}$, same as before. We then approximate (8) using posterior mode $\hat{\theta}_i^{1,j}$ and $\hat{\theta}_i^{0,j}$, written as

$$\hat{\mathbb{P}}\left(y(e) = 1 \mid \boldsymbol{x}\right) = \frac{\prod_{j=1}^{J} \hat{\theta}_{x_j}^{1,j} \mathbb{P}\left(y(e) = 1\right)}{\prod_{j=1}^{J} \hat{\theta}_{x_j}^{1,j} \mathbb{P}\left(y(e) = 1\right) + \prod_{j=1}^{J} \hat{\theta}_{x_j}^{0,j} \mathbb{P}\left(y(e) = 0\right)}.$$

This estimation is known as Maximum a posteriori estimation. Now we can write (6) as

$$\arg\max_{\boldsymbol{x}} \frac{\prod_{j=1}^{J} \hat{\eta}^{j}_{x_j}}{\prod_{j=1}^{J} \hat{\eta}^{j}_{x_j} + \frac{\mathbb{P}(y(e)=0)}{\mathbb{P}(y(e)=1)}}, \tag{12}$$

where $\hat{\eta}^{j}_{x_j} = \frac{\hat{\theta}^{1,j}_{x_j}}{\hat{\theta}^{0,j}_{x_j}}$. Since $\hat{\eta}^{j}_{i} \geq 0$, the objective in (12) is monotone increasing with $\hat{\eta}^{j}_{x_j}, \forall j$. Therefore we can solve (12) by maximizing $\prod_{j=1}^{J} \hat{\eta}^{j}_{x_j}$ over $\boldsymbol{x}$. Since $\hat{\eta}^{j}_{x_j}$ are independent across j by the assumption of the Naïve Bayes model, we can decompose the objective across j and maximize $\hat{\eta}^{j}_{x_j}$ by setting $x_j = \arg\max_i \hat{\eta}^{j}_{i}, \forall j$. The property of decomposition across j in this formulation makes the optimization problem a lot easier and requires much less computation effort than the POOL-MINLP approach. We summarize the algorithm below:

Algorithm 2. (Greedy Algorithm for POOL-MAP)

```
Require: Inputs $M, J, K_j, j = 1, \ldots, J$, dataset $\mathcal{D} = \{(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^N, y^N)\}$ and parameters for the prior distributions $\boldsymbol{\alpha}^{y,j}$.
1: $S \leftarrow \emptyset$
2: Calculate posterior distribution of $\boldsymbol{\theta}^{1,j} \sim \text{Dirichlet}(\boldsymbol{\theta}^{1,j} \mid \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D}, y(\boldsymbol{x}) = 1\})$.
3: for $m = 1$ to $M$ do
4:     Calculate posterior distribution of $\boldsymbol{\theta}^{0,j} \sim \text{Dirichlet}(\boldsymbol{\theta}^{0,j} \mid \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D}, y(\boldsymbol{x}) = 1\} \cup S)$.
5:     Calculate posterior mode $\hat{\theta}^{1,j}_{i}$ and $\hat{\theta}^{0,j}_{i}$ for $\forall i \in K_j, j = 1, \ldots, J$, and $\hat{\eta}^{j}_{i} = \frac{\hat{\theta}^{1,j}_{i}}{\hat{\theta}^{0,j}_{i}}$.
6:     Construct $\boldsymbol{x}$ by setting $x_j = \arg\max_i \hat{\eta}^{j}_{i}, \forall j$, and find the corresponding peptide $e$.
7:     $S \leftarrow (S, e)$.
8: end for
```

1.4 How POOL Combats Prediction Errors

We have compared POOL with a closely related benchmark method in the main text, the "predict-then-optimize" method, which uses the same prediction model as POOL. Suppose the prediction model is highly accurate, then the "predict-then-optimize" method should have a comparable performance v.s. POOL, or even better. However, in the benchmark experiments we have shown in the main text, POOL has a far superior performance than "predict-then-optimize". This is because our prediction model is not accurate for the prediction, and POOL is better in combating the prediction error while still giving good recommendations.

In this section, we use a simple example to explain how POOL combats the potential prediction errors. Suppose there are three peptides, A, B and C; A and B are very similar to each other, and their predicted probability being a hit are both 0.9, written as $\mathbb{P}(A \text{ is a hit}) = \mathbb{P}(B \text{ is a hit}) = 0.9$, while C is different from A or B, and C has a lower predicted probability being a hit, say $\mathbb{P}(C \text{ is a hit}) = 0.8$. We now want to find a hit from these three peptides, and due to budget limit, we could only test two peptides.

The "predict-then-optimize" approach will propose to test A and B, because they have higher predicted probabilities being a hit. However, choosing A and B does not necessarily obtain the highest probability-of-improvement. To demonstration this point, we first assume the correlation between A being a hit and B being a hit is 1, A being a hit and B being a hit are both independent from C being a hit, and $f(A), f(B), f(C)$ are all greater than b. We compute probability-of-improvement of choosing A and B, that is

$$\begin{aligned}\mathrm{PI}(\{A,B\}) &= \mathbb{P}(A \text{ is a hit}, B \text{ is a hit}) + \mathbb{P}(A \text{ is a hit}, B \text{ is not a hit}) + \mathbb{P}(A \text{ is not a hit}, B \text{ is a hit}),\\ &= 0.9 + 0 + 0,\\ &= 0.9.\end{aligned}$$

However, probability-of-improvement of choosing A and C is

$$\begin{aligned}\mathrm{PI}(\{A,C\}) &= \mathbb{P}(A \text{ is a hit}, C \text{ is a hit}) + \mathbb{P}(A \text{ is a hit}, C \text{ is not a hit}) + \mathbb{P}(A \text{ is not a hit}, C \text{ is a hit}),\\ &= 0.9 \times 0.8 + 0.9 \times 0.2 + 0.1 \times 0.8,\\ &= 0.98.\end{aligned}$$

Similarly, $\mathrm{PI}(\{B,C\}) = 0.98$. Therefore, POOL, which recommends peptides that maximize probability-of-improvement, will choose either $\{A,C\}$ or $\{B,C\}$. Now suppose A (or B) turns out to be a miss after the experiment, then B (or A) is unlikely to be a hit either because A and B are very similar to each other. On the other hand, since C is different than A and B, there might be a chance that C is a hit even if A or B is not a hit. Therefore, choosing diversified and good peptides to test, i.e., choosing $\{A,C\}$ or $\{B,C\}$ in this example, is a better strategy than choosing peptides with high predicted value regardless of diversity, i.e., choosing $\{A,B\}$. This diversity is the key to finding hits when the prediction model is not accurate.

1.5 Predicting Multiple Activities

In our application, we are looking for peptides satisfying multiple criteria. Since the experimental setup allows us to observe whether a peptide satisfies for each individual criterion, instead of simply using hits that satisfy all criteria as our data, we build seperate models for predicting each individual criterion, that is, we predict $y_i(e)$ for a peptide e, where $y_i(e)$ is 1 if the peptide satifies ith criterion, or zero otherwise. Then $\prod_i y_i(e)$ tells whether the peptide e is a hit that satisfies all criteria or not. This approach allows us to make full use of all observational data we can obtain, and is also flexible suppose we were to look for a different combination of criteria.

We then need to rewrite (6) as

$$\operatorname*{arg\,max}_{e \in E \setminus S, f(e) > b} \mathbb{P}(y_i(e) = 1, \forall i \mid \prod_i y_i(e') = 0, \forall e' \in S).$$

This objective is hard to compute because the number of configurations of $y_i(e'), \forall e \in S$ that satisfy $\prod_i y_i(e') = 0$ grows exponentially with the size of S, and the computation quickly becomes infeasible as size of S grows. We adopt a heuristic approach, which is to approximate the objective by

$$\mathbb{P}\left(y_i(e) = 1, \forall i \mid y_i(e'), \forall i, \forall e' \in S\right),$$

and we choose $y_i(e')$ as long as they satisfy $\prod_i y_i(e') = 0, \forall e' \in S$.

2 Calibration of the Experimental Result

Although the POOL method takes binary response as input, the experimental output is a metric, and more specifically, the output is the measurement of light intensity that indicates how strong the peptide labeling occurs. Therefore, we need to perform data preprocessing before feeding into POOL. However, the technical difficulty arises when the experiments were performed under different light condition. Indeed, we observed that the measurements of the same peptide in different experimental condition undergo scaling and shifting variation. To develop a consistent protocol in converting the scalar measurements to binary response, we need to first calibrate the measurements from different experiments so that they have the same scale. We propose a calibration method in this section.

Suppose we have conducted J rounds of the experiments, and tested a total of I different peptides, where some of the peptides may be tested in multiple rounds. There are a total of K kinds of treatments, and we measure the peptides under every treatment. For example, in our application, there are four treatments: 1. Sfp ($k = 1$), 2. Sfp + AcpH ($k = 2$), 3. AcpS ($k = 3$), and 4. AcpS + AcpH ($k = 4$). We use y_{ijk} to denote a measurement for ith peptide in jth round under k treatment, and assume the underlying truth value is θ_{ik}. Note that θ_{ik} does not index over j because it should be the same across rounds, and we assume the observation y_{ijk} is a result of scaling and shifting of the truth value θ_{ik} with some noise. The formulation is

$$y_{ijk} = \sigma_{jk} \cdot (\theta_{ik} + \epsilon_{ijk}) + \mu_{jk},$$

where σ_{jk} is the scaling factor, μ_{jk} is the shifting factor, and ϵ_{ijk} is i.i.d. noise drawn from $\mathcal{N}(0, a^2)$ with some constant a. In our application, there are additional constraints: $\theta_{ik_1} - \theta_{ik_2} \geq 0$ where k_1 corresponds to Sfp (or AcpS) and k_2 corresponds to Sfp + AcpH (or AcpS + AcpH), because light intensity in labeling process should be greater or equal to unlabeling process. We also tested a few "controls", which are known to be able to labeled by Sfp or AcpS, while not unlabeled by AcpH. Now we can write down log-likelihood of the data:

$$\begin{aligned} -\,logL &= \frac{1}{2a^2}\Sigma_{ijk}\left(\frac{y_{ijk} - \mu_{jk}}{\sigma_{jk}} - \theta_{ik}\right)^2 + \frac{N}{2}log(2\pi a), \\ s.t.\ \ &\forall i, \theta_{i1} - \theta i2 \geq 0, \theta_{i3} - \theta_{i4} \geq 0. \end{aligned} \tag{13}$$

To obtain the maximum likelihood estimate of the unknown parameters, we minimize (13). With a bit simplification, we want to solve the following problem:

$$\begin{aligned} \min_{\mu_{jk}, \sigma_{jk}, \theta_{ik}} \ & \Sigma_{ijk}\left(\frac{y_{ijk} - \mu_{jk}}{\sigma_{jk}} - \theta_{ik}\right)^2, \\ s.t.\ \ & \theta_{ik_1} - \theta_{ik_2} \geq 0, \\ & \theta_{ik_3} - \theta_{ik_4} \geq 0, \\ & \theta_{ik_1} - \theta_{ik_2} = 0 \ \forall i \in I_1, \\ & \theta_{ik_3} - \theta_{ik_4} = 0 \ \forall i \in I_2, \\ & \Sigma_i \theta_i = 1, \end{aligned}$$

where $I_1 =$ controlled peptides that known to be labeled by Sfp and not unlabeled by AcpH$\}$, $I_2 =$ controlled peptides that known to be labeled by AcpS and not unlabeled by AcpH$\}$, and the last equality is to avoid θs shrinking to zero.

This problem is in fact a quadratic program, and can be solved efficiently by an off-the-shelf quadratic program solver. The θ_{ik} in the solution is the calibrated value for each tested peptide.

3 General materials and methods

4 Lead peptides with fluorophores

5 Sequence alignment of lead peptides

Part I
Supplementary Figures

Part II
Supplementary Tables

class	1	2	3	4	5	6	7	8
Amino Acid	DE	NQ	FWY	RHK	AILMV	GP	ST	C

Table 1: Reduced Amino Acid Class

References

[1] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[2] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 86(1):97–106, 1964.

[3] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

[4] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[5] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[6] H. W. H. W. Sorenson. *Parameter estimation : principles and problems.* New York : Marcel Dekker, 1980. Includes bibliographies and index.