SPreCC Models

Will Nickols, Benjamin Tang, Jorge Guerra

August 23, 2022

We will present four models for different types of crystallization information. The first will be a model for predicting the presence/absence status of chemical conditions such as sodium chloride or tris. The second will be a model for predicting categorical values, which will be applied to predicting the optimal crystallization technique. The third will be a model for predicting continuous variables like the molarity of sodium chloride, the percent volume per volume of dimethyl sulfoxide, or the pH. The fourth will be a model for simultaneously predicting concentrations and polymer lengths for chemical conditions like polyethylene glycol (PEG). A single condition might use multiple models; for example, sodium chloride uses both the presence/absence model for predicting whether it should be in the crystallization mixture and the concentration model for predicting its optimal concentration if it is present. Sequence similarities will be determined using Mash, and "the database" will refer to proteins in the train set (see data curation) with the relevant data available.

1 Model 1 (presence/absence)

Let y be the true binary value of the protein of interest (such as whether or not sodium chloride is present). Let our prediction for the binary value be $\hat{y}$ such that

$$\hat{y} = \frac{\bar{x}}{n_p + 1} + \frac{n_p}{n_p + 1} \cdot \frac{\sum_i x_i \sigma(w_1 s_i + w_0)}{\sum_i \sigma(w_1 s_i + w_0)} \text{ for } i \text{ such that } p_i < \tau$$

where x_i is the binary value for protein i (e.g. whether sodium chloride was present for protein i), $s_i \in [0, 1]$ is 1 minus the Mash distance between protein i and the protein of interest, σ is the sigmoid function, p_i is the Mash p-value of the similarity between protein i and the target protein (how likely the two proteins are to have their reported degree of similarity by chance), $\bar{x}$ is the average value of the binary variable across all the database, and n_p is the total number of proteins with Mash p-values less than a threshold τ. Intuitively, we are taking a weighted average between the binary values from all the proteins and the binary values from related proteins. This ensures that the model still gives an estimate for proteins with no similar proteins in the database while allowing predictions for proteins with even a few similar proteins to be mostly determined by the conditions of those similar proteins. Within the term corresponding to similar proteins, $\sigma(w_1 s_i + w_0)$ is the weight for the binary value of protein i, and the denominator normalizes the calculation. Each weight should be some value between 0 and 1, and we expect greater sequence identities to correspond to heavier weights. However, the model allows flexibility in how much some amount of additional sequence identity should increase the weight or whether it should increase the weight at all. This weighting scheme allows much more flexibility and speed than, for example, incorporating the distance of every protein or ranking the most similar proteins. By allowing a variable number of inputs, the model avoids a need for as many independent weights as there are proteins, and it allows the weight to be determined directly from

the sequence similarity, likely a more relevant input than some ranked or positionally identified list. We will attempt to minimize the negative log-likelihood loss with regularization:

$$L(y, \hat{y}, \beta) = -[y \ln(\hat{y}) + (1-y)\ln(1-\hat{y})] + \beta||\boldsymbol{w}||^2$$

The specified model enables the fitting of two parameters: w_0 and w_1. Let $\sigma_i = \sigma(w_1 s_i + w_0)$. Applying the chain rule, we obtain the following:

$$\begin{aligned}
\frac{\partial L(\hat{y}, y, \beta)}{\partial w_0} &= \frac{\partial L(\hat{y}, y, \beta)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_0} + 2\beta w_0 \\
&= -\left[\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right] \cdot \frac{\left(\sum_i \sigma_i\right)\left(\sum_i x_i\sigma_i(1-\sigma_i)\right) - \left(\sum_i x_i\sigma_i\right)\left(\sum_i \sigma_i(1-\sigma_i)\right)}{\left(\sum_i \sigma_i\right)^2} + 2\beta w_0 \\
\frac{\partial L(\hat{y}, y, \beta)}{\partial w_1} &= \frac{\partial L(\hat{y}, y, \beta)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1} + 2\beta w_1 \\
&= -\left[\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right] \cdot \frac{\left(\sum_i \sigma_i\right)\left(\sum_i x_i\sigma_i(1-\sigma_i)s_i\right) - \left(\sum_i x_i\sigma_i\right)\left(\sum_i \sigma_i(1-\sigma_i)s_i\right)}{\left(\sum_i \sigma_i\right)^2} + 2\beta w_1
\end{aligned}$$

Because of the memory requirements involved in manipulating all the amino acid identity scores at once, we will use stochastic gradient descent to pick a protein at random, determine its amino acid identity against all the proteins in the database, predict its binary value, and update the weights according to the loss. With a learning rate α, the update statements will be as follows:

$$\begin{aligned}
w_0 &\leftarrow w_0 - \alpha \frac{\partial L(\hat{y}, y)}{\partial w_0} \\
w_1 &\leftarrow w_1 - \alpha \frac{\partial L(\hat{y}, y)}{\partial w_1}
\end{aligned}$$

To achieve an initially plausible weight scheme, the following initializations will be used: $w_0 = -1$, $w_1 = 3$.

The learning rate will start as $\alpha = 0.1$ and will decay by $1/((\text{number of proteins}) \cdot (\text{number of epochs}))$ after each update. The Mash p-value threshold will be $\tau = 1 - (1 - 1/1000)^{1/\text{number of proteins}} \approx 10^{-8}$, a level chosen so that the probability of the target protein spuriously matching any protein in the database is less than 0.001.

2 Model 2 (classification)

Let y be the true multi-class value for the protein of interest (such as whether a protein requires vapor diffusion, lipidic cubic phase, etc.) where y is a one-hot vector with each element representing one class. Let our prediction for the multiclass value be $\hat{y}$ such that

$$\hat{y} = \frac{\bar{x}}{n_p + 1} + \frac{n_p}{n_p + 1} \cdot \frac{\sum_i x_i \sigma(w_1 s_i + w_0)}{\sum_i \sigma(w_1 s_i + w_0)} \text{ for } i \text{ such that } p_i < \tau$$

where x_i is the one-hot encoded vector for protein i, $s_i \in [0, 1]$ is 1 minus the Mash distance between protein i and the protein of interest, σ is the sigmoid function, p_i is the Mash p-value of the similarity between protein i and the target protein, $\bar{x}$ is the element-wise average of the one-hot encoded vectors across all the database, and n_p is the total number of proteins with Mash p-values less than τ. Intuitively, we are taking a weighted average between the classes of all the proteins and the classes of related proteins. This ensures that the model still predicts a class for proteins with no similar proteins in the database while also allowing predictions for proteins with even a few similar proteins to be mostly determined by the classes of those similar proteins. Within the term corresponding to similar proteins, $\sigma(w_1 s_i + w_0)$ is the weight for the class of protein i, and the denominator normalizes the calculation. We will attempt to minimize the negative log-likelihood loss with regularization:

$$L(y, \hat{y}, \beta) = -\sum_{k=1}^{K} y_k \ln(\hat{y}_k) + \beta ||\boldsymbol{w}||^2$$

where K is the number of classes.

The specified model enables the fitting of two parameters: w_0 and w_1. Because of the loss specification, the gradient with respect to w_0 or w_1 will only pass through the chain rule with the $\hat{y}_k$ that corresponds to the correct y_k. Let $\sigma_i = \sigma(w_1 s_i + w_0)$. Let k be such that $y_k = 1$. Applying the chain rule, we obtain the following:

$$\begin{aligned}
\frac{\partial L(\hat{y}, y, \beta)}{\partial w_0} &= \frac{\partial L(\hat{y}, y, \beta)}{\partial \hat{y_k}} \cdot \frac{\partial \hat{y_k}}{\partial w_0} + 2\beta w_0 \\
&= -\left[\frac{1}{\hat{y_k}}\right] \cdot \frac{\left(\sum_i \sigma_i\right)\left(\sum_i x_{k,i}\sigma_i(1-\sigma_i)\right) - \left(\sum_i x_{k,i}\sigma_i\right)\left(\sum_i \sigma_i(1-\sigma_i)\right)}{\left(\sum_i \sigma_i\right)^2} + 2\beta w_0 \\
\frac{\partial L(\hat{y}, y, \beta)}{\partial w_1} &= \frac{\partial L(\hat{y}, y, \beta)}{\partial \hat{y_k}} \cdot \frac{\partial \hat{y_k}}{\partial w_1} + 2\beta w_1 \\
&= -\left[\frac{1}{\hat{y_k}}\right] \cdot \frac{\left(\sum_i \sigma_i\right)\left(\sum_i x_{k,i}\sigma_i(1-\sigma_i)s_i\right) - \left(\sum_i x_{k,i}\sigma_i\right)\left(\sum_i \sigma_i(1-\sigma_i)s_i\right)}{\left(\sum_i \sigma_i\right)^2} + 2\beta w_1
\end{aligned}$$

Because of the memory requirements involved in manipulating all the amino acid identity scores at once, we will use stochastic gradient descent to pick a protein at random, determine its amino acid identity against all the proteins in the database, predict its class, and update the weights according to the loss. With a learning rate α, the update statements will be as follows:

$$\begin{aligned}
w_0 &\leftarrow w_0 - \alpha \frac{\partial L(\hat{y}, y)}{\partial w_0} \\
w_1 &\leftarrow w_1 - \alpha \frac{\partial L(\hat{y}, y)}{\partial w_1}
\end{aligned}$$

To achieve an initially plausible weight scheme, the following initializations will be used: $w_0 = -1$, $w_1 = 3$. The learning rate will start as $\alpha = 0.1$ and will decay by $1/((\text{number of proteins}) \cdot (\text{number of epochs}))$ after each update. The Mash p-value threshold will be $\tau = 1 - (1 - 1/1000)^{1/\text{number of proteins}} \approx 10^{-8}$.

3 Model 3 (continuous, one variable)

Let y be the true value for the continuous variable of the protein of interest (such as the concentration of sodium chloride). We want to create a probability density function $f(\hat{y})$ that models the probability of the variable's true value being equal to some potential $\hat{y}$. We want to maximize the probability assigned to some small interval around the value of the true value, $\int_{y(1-\delta)}^{y(1+\delta)} f(\hat{y})d\hat{y}$ for some small δ, or, equivalently, we want to minimize the area of the fit density function that falls outside of that interval, $1 - \int_{y(1-\delta)}^{y(1+\delta)} f(\hat{y})d\hat{y}$. Because many of the variables are right skewed (and non-zero since the zero/non-zero case is dealt with by the presence/absence model), we use an interval with bounds multiplicatively rather than additively dependent on y. The two exceptions are that pH and temperature will depend additively on δ since they are non-skewed and can be zero, but the model is otherwise equivalent. This probability density can be created by applying a Gaussian kernel to a set of known values of the variable $\mathbf{x}$ from similar proteins. With x_i as the value of the crystallization variable for protein i, p_i as the Mash p-value of the similarity between protein i and the target protein, h_i as the bandwidth of the kernel element for protein i, n_p as the total number of proteins with Mash p-values less than τ, $\bar{x}$ as the average of the variable of interest over all proteins in the database, and η as the standard deviation of the variable of interest over all proteins in the database, this density function can be written as follows.

$$f(\hat{y}) = \frac{1}{n_p + 1}\frac{1}{\sqrt{2\pi}} \cdot \frac{1}{\eta} \exp\left[-\frac{\left(\frac{\hat{y}-\bar{x}}{\eta}\right)^2}{2}\right] + \frac{n_p}{n_p + 1} \cdot \frac{1}{n_p\sqrt{2\pi}} \sum_i \frac{1}{h_i} \exp\left[-\frac{\left(\frac{\hat{y}-x_i}{h_i}\right)^2}{2}\right] \text{ for } i \text{ such that } p_i < \tau \tag{1}$$

Intuitively, this is a kernel density estimate combining the distribution of the variable for all proteins and the distribution of the variable for only proteins similar to the protein of interest. However, two issues arise: not all of these similar proteins are equally similar, so they should not be weighted equally, and the optimal bandwidth h_i of each term is unknown. Both of these issues can be solved simultaneously by allowing h_i to be a function of s_i, the sequence identity (specifically, 1 minus the Mash distance between protein i and the protein of interest). This function should be continuous and likely decreasing on $[0, 1]$ because more similar proteins likely should have a smaller bandwidth for their kernels. Additionally, the function should have a codomain of $(0, \infty)$ because all the weights should be positive, but some could be much larger than others. Therefore, the relationship between h_i and s_i will be given by

$$h_i = \frac{c\sigma(w_1 s_i + w_0)}{\int_0^1 \sigma(w_1 x + w_0)dx} = \frac{c\sigma(w_1 s_i + w_0)}{\frac{\ln(e^{-w_1} + e^{w_0}) - \ln(1 + e^{w_0})}{w_1} + 1} \tag{2}$$

where σ is the sigmoid function and c is a scaling value. Letting $\mathbf{x}$ be the vector of values for similar proteins and $\mathbf{s}$ be the vector of sequence identities of the similar proteins, define the loss as

$$L(y, \mathbf{x}, \mathbf{s}, \bar{x}, \eta, \delta, \beta) = 1 - \int_{y(1-\delta)}^{y(1+\delta)} f(\hat{y})d\hat{y} + \beta(\eta - c)^2 + \beta||\boldsymbol{w}||^2$$

with f and h_i as defined above. We choose to regularize c against η because a naive bandwidth should be about the standard deviation of the variable's whole observed range, not 0. In practice, letting c vary often causes the model to break down because values of x_i close to y generate an extremely steep gradient for c, pushing c very close to 0. By creating extremely sharp peaks of density at each x_i, this undermines the effort to create a smooth probability density and makes numerical integration practically impossible. Thus, we will fix c at η. While this forces the average bandwidth to be the standard deviation of the variable, the function is capable of becoming much larger near 0 than near 1, and protein similarities near 0 often do not pass the p-value threshold for inclusion. Thus, in practice, bandwidths can become as small as necessary even with a fixed c. Still, for generality, we will treat c as a variable.

(In the implementation, all values y and x of the variable will be divided by the database mean to account for the considerable differences in scale between variables while maintaining the general right skewedness and positive values. When predicting values or ranges on the original scale, we can simply generate an estimate on this altered scale and multiply by the variable's mean.)

The specified model enables the fitting of three parameters: w_0, w_1, and c. Let h_i be as described above. Let $\sigma_i = \sigma(w_1 s_i + w_0)$. Let U be the σ normalization term $\int_0^1 \sigma(w_1 x + w_0)dx = \frac{\ln(e^{-w_1}+e^{w_0})-\ln(1+e^{w_0})}{w_1} + 1$. Let $d_i = y - x_i$. Let $z_i = \left(\frac{d_i}{h_i}\right)$. Let $m = e^{w_0} + e^{-w_1}$. Applying the chain rule, we obtain the following:

$$
\begin{aligned}
\frac{\partial f(\hat{y})}{\partial w_0} &= \sum_i \frac{\partial f(\hat{y})}{\partial h_i}\frac{\partial h_i}{\partial w_0} \\
\frac{\partial f(\hat{y})}{\partial w_1} &= \sum_i \frac{\partial f(\hat{y})}{\partial h_i}\frac{\partial h_i}{\partial w_1} \\
\frac{\partial f(\hat{y})}{\partial c} &= \sum_i \frac{\partial f(\hat{y})}{\partial h_i}\frac{\partial h_i}{\partial c} \\
\frac{\partial f(\hat{y})}{\partial h_i} &= \frac{1}{(n_p+1)\sqrt{2\pi}}\frac{\exp(-z_i^2/2)(z_i^2-1)}{h_i^2} \\
\frac{\partial h_i}{\partial w_0} &= \frac{c\sigma_i(1-\sigma_i)}{U} - \frac{c\sigma_i(\frac{e^{w_0}}{m} - \frac{e^{w_0}}{1+e^{w_0}})}{w_1 U^2} \\
\frac{\partial h_i}{\partial w_1} &= \frac{s_i c\sigma_i(1-\sigma_i)}{U} - \frac{c\sigma_i[\frac{-w_1 e^{-w_1}}{e^{-w_1}+e^{w_0}} - (\ln(e^{-w_1}+e^{w_0}) - \ln(1+e^{w_0}))]}{w_1^2 U^2} \\
\frac{\partial h_i}{\partial c} &= \frac{h_i}{c}
\end{aligned}
$$

However, we are actually interested in the integrals of these quantities, which are obtained with the Leibniz integral rule:

$$
\begin{aligned}
\frac{\partial L(y,\mathbf{x},\mathbf{s},\bar{x},\eta,\delta,\beta)}{\partial w_0} &= -\int_{y(1-\delta)}^{y(1+\delta)} \frac{\partial f(\hat{y})}{\partial w_0} d\hat{y} + 2\beta w_0 \\
\frac{\partial L(y,\mathbf{x},\mathbf{s},\bar{x},\eta,\delta,\beta)}{\partial w_1} &= -\int_{y(1-\delta)}^{y(1+\delta)} \frac{\partial f(\hat{y})}{\partial w_1} d\hat{y} + 2\beta w_1 \\
\frac{\partial L(y,\mathbf{x},\mathbf{s},\bar{x},\eta,\delta,\beta)}{\partial c} &= -\int_{y(1-\delta)}^{y(1+\delta)} \frac{\partial f(\hat{y})}{\partial c} d\hat{y} + 2\beta(c-\eta)
\end{aligned}
\tag{3}
$$

Because of the memory requirements involved in manipulating all the amino acid identity scores at once, we will use stochastic gradient descent to pick a protein at random, determine its amino acid identity against

the proteins in the database, compute its density function, and update the weights according to the loss. With a learning rate α, the update statements will be as follows:

$$\begin{aligned} w_0 &\leftarrow w_0 - \alpha \frac{\partial L(y, \mathbf{x}, \mathbf{s}, \bar{x}, \eta, \delta, \beta)}{\partial w_0} \\ w_1 &\leftarrow w_1 - \alpha \frac{\partial L(y, \mathbf{x}, \mathbf{s}, \bar{x}, \eta, \delta, \beta)}{\partial w_1} \\ c &\leftarrow c - \alpha \frac{\partial L(y, \mathbf{x}, \mathbf{s}, \bar{x}, \eta, \delta, \beta)}{\partial c} \end{aligned} \tag{4}$$

The partial derivative of the density function with respect to each parameter will be computed exactly, but the Leibniz integrals will be approximated with a left Riemann sum and a $\Delta\hat{y}$ of $y/100$.

By linearity of expectation, the expectation of f is simply

$$\frac{1}{n_p + 1}\bar{x} + \frac{n_p}{n_p + 1}\frac{1}{n_p}\sum_i x_i \text{ for } i \text{ such that } p_i < \tau$$

The mode of the distribution can be found by evaluating the probability density function (PDF) from $\min(\bar{x}, \min(\{x_i | x_i \in \mathbf{x}\}))$ to $\max(\bar{x}, \max(\{x_i | x_i \in \mathbf{x}\}))$ with a step size of the difference divided by 1,000 and recording the value of the variable at the maximum value of the PDF. Because the density of an individual kernel input decreases on either side of its mean, the mode is guaranteed to be between these bounds. Because numeric integration over a large sum of variables is computationally costly, we can find an estimated 95% confidence interval from the 2.5^{th} percentile to the 97.5^{th} percentile of the kernel density by iterating from either end of the distribution. For the 2.5^{th} percentile, we begin iterating upwards from $\hat{y} = \max(\min(\text{value of the variable for all proteins in the database}), \min(\Phi^{-1}(0.025)\eta + \bar{x}, \{\min(\Phi^{-1}(0.025)h_i + x_i) | x_i \in \mathbf{x}\}))$ and taking steps of the same size as for the mode until

$$\frac{1}{n_p + 1}\Phi\left(\frac{\hat{y} - \bar{x}}{\eta}\right) + \frac{n_p}{n_p + 1} \cdot \frac{1}{n_p}\sum_i \Phi\left(\frac{\hat{y} - x_i}{h_i}\right) \geq 0.025$$

with Φ denoting the standard normal cumulative density function. We take this $\hat{y}$ as the 2.5^{th} percentile. Likewise, for the 97.5^{th} percentile, we begin iterating downwards from

$$\hat{y} = \min(\max(\text{value of the variable for all proteins in the database}), \max(\Phi^{-1}(0.975)\eta + \bar{x}, \max(\{\Phi^{-1}(0.975)h_i + x_i) | x_i \in \mathbf{x}\}))$$

and taking steps of the same size until

$$\frac{1}{n_p + 1}\Phi\left(\frac{\hat{y} - \bar{x}}{\eta}\right) + \frac{n_p}{n_p + 1} \cdot \frac{1}{n_p}\sum_i \Phi\left(\frac{\hat{y} - x_i}{h_i}\right) < 0.975$$

We then take the $\hat{y}$ before the current one (the last one where the expression was greater than 0.975) as the 97.5^{th} percentile. As a proof sketch that the minimum of the kernel elements' 2.5^{th} percentiles is less than or equal to the 2.5^{th} percentile of the whole kernel density, consider that for any other term besides this minimizer, that term's 2.5^{th} percentile must be larger by construction, so it contributes more density to the kernel above its own 2.5^{th} percentile and therefore above the minimum 2.5^{th} percentile than below either. Therefore, this other term shifts the density of the total kernel density upwards, adding more weight above the minimum 2.5^{th} percentile, guaranteeing that the overall 2.5^{th} percentile is larger than the minimum of

the individual kernel elements' 2.5^{th} percentiles. The proof for the 97.5^{th} percentile is analogous. Further bounding the search range by the minimum and maximum values of the variable in the database ensures that a protein with only a few distantly related proteins does not require a massive search space due to very large bandwidths in the kernel density.

To achieve an initially plausible bandwidth scheme, the following initializations will be used: $w_0 = -1$, $w_1 = -2$, $c = \eta$.

The learning rate will start as $\alpha = 0.1$ and will decay by $1/((\text{number of proteins}) \cdot (\text{number of epochs}))$ after each update. The Mash p-value threshold will be $\tau = 1 - (1 - 1/1000)^{1/\text{number of proteins}} \approx 10^{-8}$.

4 Model 4 (continuous, two variables)

Let (y_1, y_2) be the true value for the two-dimensional continuous variable of the protein of interest (such as the concentration and length of PEG). We want to create a probability density function $f(\hat{y_1}, \hat{y_2})$ that models the probability of the variable's true value being equal to some potential $(\hat{y_1}, \hat{y_2})$. We want to maximize the probability assigned to some small interval around the true value of the variable, $\int_{y_1(1-\delta)}^{y_1(1+\delta)} \int_{y_2(1-\delta)}^{y_2(1+\delta)} f(\hat{y_1}, \hat{y_2}) d\hat{y_1} d\hat{y_2}$ for some small δ, or, equivalently, we want to minimize the area of the fit density function that falls outside of that interval, $1 - \int_{y_1(1-\delta)}^{y_1(1+\delta)} \int_{y_2(1-\delta)}^{y_2(1+\delta)} f(\hat{y_1}, \hat{y_2}) d\hat{y_1} d\hat{y_2}$. This probability density can be created by applying a Gaussian kernel to a set of known values of the variable $\mathbf{x}$ from similar proteins that contain the variable of interest. With $(x_{1,i}, x_{2,i})$ as the value of the variable for protein i, p_i as the Mash p-value of the similarity between protein i and the target protein, $(h_{1,i}, h_{2,i})$ as the bandwidths of the kernel element for protein i, n_p as the total number of proteins with Mash p-values less than τ, $(\bar{x_1}, \bar{x_2})$ as the element-wise average of the variable of all proteins in the database, and (η_1, η_2) as the standard deviations of the variable of all proteins in the database, this density function can be written as follows.

$$f(\hat{y_1}, \hat{y_2}) = \frac{1}{n_p + 1} \frac{1}{2\pi} \cdot \frac{1}{\eta_1 \eta_2} \exp\left[-\frac{\left(\frac{\hat{y_1} - \bar{x_1}}{\eta_1}\right)^2 + \left(\frac{\hat{y_2} - \bar{x_2}}{\eta_2}\right)^2}{2} \right]$$
$$+ \frac{1}{n_p + 1} \cdot \frac{1}{2\pi} \sum_i \frac{1}{h_{1,i} h_{2,i}} \exp\left[-\frac{\left(\frac{\hat{y_1} - x_{1,i}}{h_{1,i}}\right)^2 + \left(\frac{\hat{y_2} - x_{2,i}}{h_{2,i}}\right)^2}{2} \right] \text{ for } i \text{ such that } p_i < \tau$$

Intuitively, this is a kernel density estimate weighing together the distribution of the variable for all proteins and the distribution of the variable for only proteins similar to the protein of interest. However, two issues arise: not all of these similar proteins are equally similar, so they should not be weighted equally, and the optimal bandwidths $(h_{1,i}, h_{2,i})$ of each term are unknown. Both of these issues can be solved simultaneously by allowing $(h_{1,i}, h_{2,i})$ to be a function of s_i, the sequence identity (specifically, 1 minus the Mash distance between protein i and the protein of interest). This function should be continuous and decreasing on $[0, 1]$ because more similar proteins likely should have a smaller bandwidth for their kernels. Additionally, the function should have a codomain of $(0, \infty)$ because all the weights should be positive, but some could be much larger than others. Therefore, the relationship between $(h_{1,i}, h_{2,i})$ and s_i will be given by

$$h_{j,i} = \frac{c_j \sigma(w_{j,1} s_i + w_{j,0})}{\int_0^1 \sigma(w_{j,1} x + w_{j,0}) dx} = \frac{c_j \sigma(w_{j,1} s_i + w_{j,0})}{\frac{\ln(e^{-w_{j,1}} + e^{w_{j,0}}) - \ln(1 + e^{w_{j,0}})}{w_{j,1}} + 1} \tag{5}$$

for $j \in \{1, 2\}$ where σ is the sigmoid function and c_j is a scaling value. Letting $\mathbf{x}$ be the vector of values for similar proteins and $\mathbf{s}$ be the vector of sequence identities of the similar proteins, define the loss as

$$L((y_1, y_2), \mathbf{x}, \mathbf{s}, (\bar{x_1}, \bar{x_2}), (\eta_1, \eta_2), (\delta_1, \delta_2), \beta) = 1 - \int_{y_1(1-\delta)}^{y_1(1+\delta)} \int_{y_2(1-\delta)}^{y_2(1+\delta)} f(\hat{y_1}, \hat{y_2}) d\hat{y_1} d\hat{y_2} + \beta ||\boldsymbol{\eta} - \mathbf{c}||^2 + \beta ||\boldsymbol{w}||^2$$

with f and $(h_{1,i}, h_{2,i})$ as defined above. We choose to regularize $\mathbf{c}$ against $\boldsymbol{\eta}$ because a naive bandwidth should be about the standard deviation of the variable, not 0. In practice, letting c_1 and c_2 vary often causes the model to break down because values of x_i close to y generate an extremely steep gradient for c_1 and c_2, pushing them very close to 0. By creating extremely sharp peaks of density at each x_i, this undermines the effort to create a smooth probability density and makes numerical integration essentially impossible. Thus, we will fix c_1 and c_2 at η_1 and η_2 respectively. While fixing these values forces the average bandwidth to be the standard deviation of all values for the variable, the function is capable of becoming much larger near 0 than near 1, and protein similarities near 0 often do not pass the p-value threshold for inclusion. Thus, in practice, bandwidths can become as small as necessary even with fixed c_1 and c_2. Still, for generality, we will treat both as variables.

(As before, all values y and x of the variable will be divided elementwise by their means.)

The specified model enables the fitting of six parameters: $w_{1,0}$, $w_{1,1}$, c_1, $w_{2,0}$, $w_{2,1}$, and c_2. Let $(h_{1,i}, h_{2,i})$ be as described above. For $j \in \{1, 2\}$, let $\sigma_{j,i} = \sigma(w_{j,1} s_i + w_{j,0})$. Let U_j be the σ_j normalization term $\int_0^1 \sigma(w_{j,1} x + w_{j,0}) dx = \frac{\ln(e^{-w_{j,1}} + e^{w_{j,0}}) - \ln(1 + e^{w_{j,0}})}{w_{j,1}} + 1$. Let $d_{j,i} = y_j - x_{j,i}$. Let $z_{j,i} = \left(\frac{d_{j,i}}{h_{j,i}}\right)$. Let $m_j = e^{w_{j,0}} + e^{-w_{j,1}}$, $r_j = \frac{e^{w_{j,0}}}{m_j} - \frac{e^{w_{j,0}}}{1 + e^{w_{j,0}}}$, $v_j = [\frac{-w_{j,1} e^{-w_{j,1}}}{m_j} - (\ln(m_j) - \ln(1 + e^{w_{j,0}}))]/w_{j,1}$. Applying the chain rule, we obtain the following:

$$\begin{aligned}
\frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial w_{j,0}} &= \sum_i \frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial h_{j,i}} \frac{\partial h_{j,i}}{\partial w_{j,0}} \\
\frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial w_{j,1}} &= \sum_i \frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial h_{j,i}} \frac{\partial h_{j,i}}{\partial w_{j,1}} \\
\frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial c_j} &= \sum_i \frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial h_{j,i}} \frac{\partial h_{j,i}}{\partial c_j} \\
\frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial h_{1,i}} &= \frac{1}{(n_p + 1) 2\pi} \frac{\exp(-z_{2,i}^2/2)}{h_{2,i}} \frac{\exp(-z_{1,i}^2/2)(z_{1,i}^2 - 1)}{h_{1,i}^2} \\
\frac{\partial f(\hat{y_1}, \hat{y_2})}{\partial h_{2,i}} &= \frac{1}{(n_p + 1) 2\pi} \frac{\exp(-z_{1,i}^2/2)}{h_{1,i}} \frac{\exp(-z_{2,i}^2/2)(z_{2,i}^2 - 1)}{h_{2,i}^2} \\
\partial h_{j,i}/\partial w_{j,0} &= [c_j \sigma_{j,i}(1 - \sigma_{j,i})]/U_j - [c_j \sigma_{j,i} r_j]/[w_{j,1}(U_j)^2] \\
\partial h_{j,i}/\partial w_{j,1} &= [s_i \cdot c_j \sigma_{j,i}(1 - \sigma_{j,i})]/U_j - [c_j \sigma_{j,i} v_j]/[w_{j,1}(U_j)^2] \\
\frac{\partial h_{j,i}}{\partial c_j} &= \frac{h_{j,i}}{c_j}
\end{aligned}$$

However, we are actually interested in the integrals of these quantities, which are obtained with the Leibniz integral rule:

$$\begin{aligned}
\frac{\partial L((y_1,y_2),\mathbf{x},\mathbf{s},(\bar{x_1},\bar{x_2}),(\eta_1,\eta_2),(\delta_1,\delta_2),\beta)}{\partial w_{j,0}} &= -\int_{y_1(1-\delta)}^{y_1(1+\delta)}\int_{y_2(1-\delta)}^{y_2(1+\delta)}\frac{\partial f(\hat{y_1},\hat{y_2})}{\partial w_{j,0}}d\hat{y_1}d\hat{y_2} + 2\beta w_{j,0} \\
\frac{\partial L((y_1,y_2),\mathbf{x},\mathbf{s},(\bar{x_1},\bar{x_2}),(\eta_1,\eta_2),(\delta_1,\delta_2),\beta)}{\partial w_{j,1}} &= -\int_{y_1(1-\delta)}^{y_1(1+\delta)}\int_{y_2(1-\delta)}^{y_2(1+\delta)}\frac{\partial f(\hat{y_1},\hat{y_2})}{\partial w_{j,1}}d\hat{y_1}d\hat{y_2} + 2\beta w_{j,1} \\
\frac{\partial L((y_1,y_2),\mathbf{x},\mathbf{s},(\bar{x_1},\bar{x_2}),(\eta_1,\eta_2),(\delta_1,\delta_2),\beta)}{\partial c_j} &= -\int_{y_1(1-\delta)}^{y_1(1+\delta)}\int_{y_2(1-\delta)}^{y_2(1+\delta)}\frac{\partial f(\hat{y_1},\hat{y_2})}{\partial c_j}d\hat{y_1}d\hat{y_2} + 2\beta(c_j-\eta_j)
\end{aligned} \tag{6}$$

Because of the memory requirements involved in manipulating all the amino acid identity scores at once, we will use stochastic gradient descent to pick a protein at random, determine its amino acid identity against the proteins in the database, compute its density function, and update the weights according to the loss. With a learning rate α, the update statements will be as follows:

$$\begin{aligned}
w_{j,0} &\leftarrow w_{j,0} - \alpha\frac{\partial L((y_1,y_2),\mathbf{x},\mathbf{s},(\bar{x_1},\bar{x_2}),(\eta_1,\eta_2),(\delta_1,\delta_2),\beta)}{\partial w_{j,0}} \\
w_{j,1} &\leftarrow w_{j,1} - \alpha\frac{\partial L((y_1,y_2),\mathbf{x},\mathbf{s},(\bar{x_1},\bar{x_2}),(\eta_1,\eta_2),(\delta_1,\delta_2),\beta)}{\partial w_{j,1}} \\
c_j &\leftarrow c_j - \alpha\frac{\partial L((y_1,y_2),\mathbf{x},\mathbf{s},(\bar{x_1},\bar{x_2}),(\eta_1,\eta_2),(\delta_1,\delta_2),\beta)}{\partial c_j}
\end{aligned} \tag{7}$$

The partial derivative of the density function with respect to each parameter will be computed exactly, but the Leibniz integrals will be approximated with a left Riemann sum and a $\Delta\hat{y_j}$ of $y_j/100$.

By linearity of expectation, the expectation of f is simply

$$\frac{1}{n_p+1}\bar{x} + \frac{n_p}{n_p+1}\frac{1}{n_p}\sum_i x_i \text{ for } i \text{ such that } p_i < \tau$$

The mode of the distribution can be found by evaluating the PDF from $\min(\bar{x}, \min(\{x_i | x_i \in \mathbf{x}\}))$ to $\max(\bar{x}, \max(\{x_i | x_i \in \mathbf{x}\}))$ with a step size of the difference divided by 1,000 and recording the value of the variable at the largest value of the PDF. Here, min indicates the element-wise minimum of the two-element vector (i.e. the minimum of $x_{j,i}$ over all i for $j = 1$ and $j = 2$ separately). Because the density of an individual kernel input decreases on all sides of its mean, the mode is guaranteed to be between these bounds. Because numeric integration over a large sum of variables is computationally costly, we can use the marginal density for each of the two elements of the variable to find an estimated 95% confidence interval from the 2.5^{th} percentile to the 97.5^{th} percentile as before.

To achieve an initially plausible bandwidth scheme, the following initializations will be used for $j \in \{1, 2\}$: $w_{j,0} = -1, w_{j,1} = -2, c_j = \eta_j$. The learning rate will start as $\alpha = 0.1$ and will decay by $1/((\text{number of proteins}) \cdot (\text{number of epochs}))$ after each update. The Mash p-value threshold will be $\tau = 1-(1-1/1000)^{1/\text{number of proteins}} \approx 10^{-8}$.