Gradient Descent for Logistic-Weighted Similarity-Based Prediction

Will Nickols

August 7, 2022

1 Model

Let y be the true value for the binary value we are interested in. 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, $s_i \in [0, 1]$ 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 alignment between protein i and the target protein, $\bar{x}$ is the average value of the binary condition across all the dataset excluding the protein of interest, 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 conditions of all the proteins and the conditions of related proteins. This ensures that the model still gives an estimate 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 conditions of those similar proteins. Within the term corresponding to similar proteins, $\sigma(w_1 s_i + w_0)$ is the weight for the crystallization condition of protein i, and the denominator normalizes the calculation. Each weight should be some value between 0 and 1 with greater sequence identities corresponding to heavier weights, but the model allows flexibility in how much some amount of additional sequence identity should increase the weight. We will attempt to minimize the negative log-likelihood loss: $L(y, \hat{y}) = -[y \ln(\hat{y}) + (1 - y) \ln(1 - \hat{y})]$.

2 Gradient

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 and the fact that for $\sigma(x)$, $\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x))$, we obtain the following:

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

3 Stochastic Gradient Descent and Model Updating

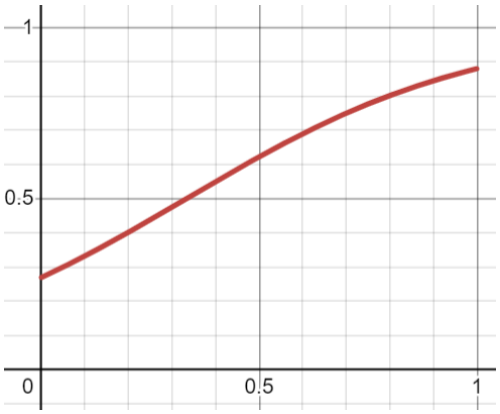
Because of the memory requirements involved in manipulating all the amino acid identity scores at once, we will use stochastic gradient descent to pick a protein at random, determine its amino acid identity against all the other proteins, predict its condition, and update the weights according to the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

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

There are a few optimizations to speed run time. First, before choosing a protein for gradient descent, we can store $n' \sum_{j=1}^{n'} x_j$ where n' is the total number of proteins. Then, once we choose a protein i, we can simply subtract x_i from this stored sum and divide by $n' - 1 = n$ to obtain the first term of $\hat{y}$. Second, all the σ_i can be calculated once and stored to allow the calculation of all the $1 - \sigma_i$, which can also be stored. These vectors then enable vectorized multiplication to obtain all the terms in the gradients, and any repeated sums between the gradient of w_0 and w_1 can be stored and reused.

4 Initialization and runtime

To achieve an initially plausible weight scheme, the following initializations will chosen: $w_0 = -1$, $w_1 = 3$. The following image shows the weights produced by these parameters.



The learning rate will start as $\alpha = 0.1$ and will decay by $1/(n' \cdot (\text{number of epochs}))$ after each update. If the weights are within a specified convergence radius for a specified number of iterations (0.1 and 20,000 by default), training for that condition's weights will end. Additionally, because the weight updating proceeds much faster than the Mash distance calculation, the Mash distances will be computed in parallel for a batch of proteins, and the weight updating will proceed sequentially. Finally, the Mash p-value threshold will be $\tau = 1/n' \approx 9 \cdot 10^{-6}$.