The Implementation and Improvement of the Hidden Markov Model for Labeled Sequences

Andrew Choi[1], Mason Bulling[2], Kaitlyn Ma[3], Alex Giang[4], and Jan Bobrowski[5]

[1]Computer Science, 2025, asc269
[2]Computer Science, 2025, mcb343
[3]Computer Science, 2025, kim32
[4]Computer Science, 2025, akg72
[5]Computer Science, 2025, jhb344

ABSTRACT

Hidden Markov Models (HMMs) play a pivotal role in various computational applications, extending their utility across a wide spectrum of disciplines. A novel variant of the HMM, known as the Class Hidden Markov Model (CHMM), is primarily designed for handling labeled observations. The distinctiveness of the CHMM lies in its approach to parameter estimation, employing a maximum likelihood method that diverges from traditional HMMs. This divergence is characterized by the shift from modeling the statistical properties of training sequences to an emphasis on optimizing recognition performance. CHMMs were introduced in a paper titled "Hidden Markov Models for Labeled Sequences", by Anders Krogh [1]. Krogh theorized the CHMM by observations through his own research, yet never placed the CHMM into practice. The goal of this project was to implement the CHMM and analyze its effectiveness as a machine learning model. The project focused on predicting the secondary protein structure of human and rat protein sequences, those of which have been mapped out through either wet lab work or other verifiable methods. The computational pipeline mainly involved model parameterization, class emission estimation, gradient descent, stochastic gradient descent, and adaptive moment estimation. A significant challenge arises in the implementation of the standard forward-backward procedure within the CHMM framework. To address this, the paper proposes an innovative 'incremental EM' method, a tailored approach that facilitates the estimation process in this new model. The model was trained on about 1000 protein sequences each for human and rat data, using a Leave-One-Out Cross Validation technique for the validation tests. The model created turned out to be not exceptional, but performed better than a random selection of labels. With gradient descent, the model predicted the sequences with 42-43% accuracy, and with stochastic gradient descent, the model predicted with 55-65% accuracy. ADAM was slightly less accurate at 50-53%, and as a control, random selection was 25% accurate. While the CHMM could prove effective in theory as proposed by Krogh, there is still investigation that needs to be done to analyze how to improve the accuracy of the model.

Keywords: Hidden Markov Model, Nucleotide, Sequence prediction, Protein, Amino Acid, Viterbi, Forward-backward algorithm

Project type: Implementation, Research

Project repository: Github Repository

1 Introduction

Hidden Markov Models are a powerful statistical framework to model sequential data with underlying probability distribution. For obvious reasons, such a model is useful for predicting a varied number of observable events - speech signals, gene finding, language decoding, among others. Under common settings, parameter estimation involves maximizing the likelihood of a given set of sequence(s) (MLE), or maximizing the likelihood of a set of sequence(s) given a prior belief (MAP). To see a figure visualizing our approach, see **Figure 1**.

For this project, the focus was to predict the secondary structure of protein sequence data. The secondary structure of protein sequences is crucial as it lays the foundation for the protein's 3D conformation, essential for biological function. Elements like alpha helices and beta sheets, formed due to hydrogen bonding patterns among amino acids, determine how the protein interacts with other molecules. This interaction specificity underpins vital processes such as enzyme catalysis, signal transduction, and molecular recognition. Misfolding in the secondary structure can lead to dysfunctional proteins, causing diseases like

Alzheimer's. Understanding secondary structures also aids in drug design, allowing scientists to create molecules that precisely interact with target proteins, enhancing therapeutic efficacy and reducing side effects. Currently, wet lab research and various methods to determine secondary structure of a protein can be expensive and time intensive. Therefore, creating a machine learning model that can accurately predict the secondary structure of a protein given its sequence can be very helpful in providing a faster and more accurate way to map a protein's sequence to its structure.

Hidden Markov Models have drawbacks in their ability to predict something like secondary protein structure. This is because of their inherent inability to include external information in the data, such as class labels. In our case, this is the class labels that represent secondary structure.

On the other hand, Class Hidden Markov Models (CHMM) are a concept theorized by Anders Krogh [1]. Krogh, through his own previous research on protein modeling and gene modeling, thought of a possibility where CHMMs could accurately predict sequential data such as protein sequences [2] [3]. CHMMs allow us to incorporate additional information in regards to amino acid sequences with corresponding class label structures in a supervised learning like fashion. Normally, HMMs have transition and emission probabilities, which represent the probabilities between hidden states and between hidden states and outputs, respectively. In order to incorporate the class labels, CHMMs introduce another probability distribution over such class labels, which we call class emission probabilities. The CHMM then has two emission probabilities, which we call symbol emission probabilities and class emission probabilities. This allows the CHMM to incorporate the secondary structure as a part of the model. Krogh details a mathematical gradient descent and stochastic gradient descent approach to the model training, which we will delve into further later. Thus, in theory, CHMMs provide a much easier way to predict secondary structure than HMMs, because of their inherent ability to include the class labels with ease. Since Krogh had only novelly theorized the CHMM, the goal of this project is to implement it correctly to see if his theory can be translated to real implementation. Then, we will analyze the results in the ability of the CHMM to accurately predict secondary structure.

2 Methods

2.1 Class Hidden Markov Models

Class Hidden Markov Models (CHMMs) require a few more definitions than the traditional Hidden Markov Model.

Below are the notional definitions that we will continue to use throughout this paper.

- Observations: $\mathbf{s} = (s_1, s_2, \dots, s_n)$
- Labels: $\mathbf{c} = (c_1, c_2, \dots, c_n)$
- $b_i(a)$: Symbol emission probabilities: the probability of emitting symbol a in state i
- a_{ij}: Transition probabilities: the probability for making a transition from state i to j
- $\phi_i(x)$: Class emission probabilities: the probability of emitting class label x in state i

2.2 Data Engineering

There are a few important things to note for obtaining our data. The actual reproducibility of the results can be found in the README file on the Github Repo. However, here are the important parts:

- The protein sequences were obtained from UniProt.org
- We obtained data that involved known proteins from only humans (Homo sapiens) and mice (Mus musculus). We chose humans and mice mainly because of their commonality and their abundance of data. For every protein we used, it was important that two things were available: the sequence itself (which almost every protein has), and the secondary structure of the protein.
- We used about 500 sequences per organism for the size of the data set. The amount of available mice data was limited to this number, and we decided to go with the same amount for humans, to keep the amount of data even as well as for time reasons (for the amount of time it took to train the model).

The nature of our data are amino acid sequences with associated class label sequences. In animals in general, there are 20 common amino acids. These 20 letters make up all of the protein sequence data that we have retrieved. Our procedure for encoding these amino acid sequences is as follows:

$$n = \text{length of sequence}$$
$$m = \text{number of amino acids}$$

We create a bijection mapping between the number of amino acids and $[1, m]$. This can be represented as a function $\mathbb{R}^n \to \mathbb{R}^n$ where the ith position corresponds to the ith amino acid mapping to $[1, m]$.

The second thing we obtained for every protein were the class label sequences. The secondary structure is mainly governed by two things: alpha helix or beta sheet. The other important piece of structure, but less so, is a turn. A turn isn't too important because they are less common, but they link the end of one element of secondary structure to the beginning of another. One important thing to note, is that some parts of sequence do not correspond to any structure. We keep track of this as well.

We do a similar procedure for the class label sequence.

$$n = \text{length of sequence}$$

We create a bijection mapping between no structure, alpha-helix, beta-sheet, turn and $[0,1,2,3]$. This can be represented as a function $\mathbb{R}^n \to \mathbb{R}^n$ where the ith class label position corresponds to the ith secondary structure mapping to $[0,1,2,3]$.

For simplicity, we aggregate class transition properties, symbol emission probabilities, and class emission probabilities into θ and ϕ respectively.

$$\theta = [a_{10},\ldots,a_{20},\ldots,a_{nn},b_1,\ldots,b_n] := \text{a vector containing these probabilities}$$
$$\phi = [\phi_1,\ldots,\phi_n] := \text{a vector containing these class label emission probabilities}$$

2.3 Parameter Estimation via Gradient Descent

We can estimate $\hat{\theta}$ by maximizing the conditional probability of observing class label sequence **c** given observation sequence **s**, state transition and symbol emission probabilities **A**, and class label emission probabilities **B** [1].

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} P(\mathbf{c}|\mathbf{s},\theta,\phi) = \underset{\theta}{\operatorname{argmax}} \frac{P(\mathbf{c},\mathbf{s}|\theta,\phi)}{P(\mathbf{s}|\theta)}$$

Since the log operator is a monotonic function, we can turn this maximization objective into a minimization objective by taking the negative log of the expression.

$$\begin{aligned} L_f &= -\log P(\mathbf{s}|\theta) \\ L_c &= -\log P(\mathbf{c},\mathbf{s}|\theta,\phi) \\ L &= -\log \frac{P(\mathbf{c},\mathbf{s}|\theta,\phi)}{P(\mathbf{s}|\theta)} = L_c - L_f \\ \hat{\theta} &= \underset{\theta}{\operatorname{argmin}}(L) \end{aligned}$$

Let $n_i(\pi,\mathbf{s})$ denote the number of times parameter θ_i is used in the path π for observation sequence **s**.

$$\begin{aligned} P(\mathbf{s}|\theta) &= \sum_{\pi} P(\mathbf{s},\pi|\theta) \\ P(\pi,\mathbf{s}|\theta) &= \prod_i \theta^{n_i(\pi,\mathbf{s})} \end{aligned}$$

Let $n_k(\mathbf{s})$ denote the expected number of times the kth parameter is used by observation **s**. Let $q_j(\pi,\mathbf{c},\mathbf{s})$ be the number of times the parameter ϕ_i is used in path π for observation sequence **s**.

$$\frac{dP(\pi,\mathbf{s}|\theta)}{d\theta_k} = \frac{n_k(\pi,\mathbf{s})}{\theta_k}\prod_i \theta_i^{n_i(\pi,\mathbf{s})} = \frac{n_k(\pi,\mathbf{s})}{\theta_k}P(\pi,\mathbf{s}|\theta)$$
$$\frac{dL_f}{d\theta_k} = -\sum_{\pi}\frac{n_k(\pi,\mathbf{s})}{\theta_k}\frac{P(\pi,\mathbf{s}|\theta)}{P(\mathbf{s}|\theta)} = -\sum_{\pi}\frac{n_k(\pi,\mathbf{s})}{\theta_k}P(\pi|\mathbf{s},\theta) = -\frac{n_k(s)}{\theta_k}$$

$$P(\pi,\mathbf{c},\mathbf{s}|\theta,\phi) = \prod_i \theta_i^{n_i(\pi,\mathbf{s})}\prod_i \phi_j^{q_j(\pi,\mathbf{c},\mathbf{s})}$$

Let $m_k(\mathbf{c},\mathbf{s})$ denote the expected number of times the kth parameter is used by the observations $\mathbf{s}$ and gives the correct labeling $\mathbf{c}$.

[1]

$$\frac{d}{d\theta_k}P(\theta,\mathbf{c},\mathbf{s}|\theta,\phi) = \frac{n_k(\pi,\mathbf{s})}{\theta_k}P(\pi,\mathbf{c},\mathbf{s}|\theta,\phi)$$

$$\frac{dL_c}{d\theta_k} = -\sum_{\pi}\frac{n_k(\pi,\mathbf{s})}{\theta_k}\frac{P(\theta,\mathbf{c},\mathbf{s}|\theta,\phi)}{P(\mathbf{c},\mathbf{s}|\theta,\phi)} = -m_k(\mathbf{c},\mathbf{s})$$

$$\frac{dL}{d\theta_k} = -\frac{m_k(\mathbf{c},\mathbf{s})-n_k(\mathbf{s})}{\theta_k}$$

We use the forward-backward algorithms to calculate m_k and n_k. n_k is calculated using the forward-backward algorithms on all paths. m_k is calculated using the forward-backward algorithm only on paths where the sequence of states aligns correctly with the observed sequence of class labels. To see how we did this, reference **Algorithm 1**.

Since L can be highly non-convex - multiple similar conditional probabilities ratios can lead to the same objective - ordinary gradient descent can lead to minimums that generalize poorly to new amino acid sequences.

2.4 Class Probability Parameter Estimation

We can estimate the class emission probabilities ϕ using Maximum Likelihood Estimation (MLE). Let $q_k(\mathbf{c},\mathbf{s})$ be the expected number of times parameter ϕ_k is used for observation sequences corresponding to label $\mathbf{c}$.

[2]

$$\hat{\phi_k} = \underset{\phi}{\operatorname{argmax}} P(\mathbf{c}|\mathbf{s},\theta,\phi) = \underset{\phi}{\operatorname{argmin}} L_c$$

$$\frac{d}{d\phi_k}P(\pi,\mathbf{c},\mathbf{s}|\theta,\phi) = \frac{q_k(\pi,\mathbf{c},\mathbf{s})}{\phi_k}\cdot P(\pi,\mathbf{c},\mathbf{s}|\theta,\phi)$$

$$\frac{dL}{d\phi_k} = \frac{-q_k(\pi,\mathbf{c},\mathbf{s})}{\phi_k}$$

2.5 Implementation

Research suggests that amino acid sequences with high G+C content are associated with more stable secondary structures [4]. So we define two hidden states corresponding to high G+C and low G+C content. Furthermore, for our theta initialization we define transition probabilities to be uniform, observed symbol emission probabilities to be uniform for each state. We do a similar procedure for the class emission probabilities as well.

Then we perform the gradient calculations as shown in [1] and [2] and perform gradient descent with some fixed step size.

Since it's highly improbable to completely estimate an amino acid sequence's secondary structure, we define accuracy as the number of correct secondary structures for every position in its predicted sequence against its true labeled sequence.

As a baseline for estimating a secondary structure sequence for a given amino acid sequence by random guessing. Furthermore, let's assume (for simplicity) that the secondary structure at each given position is independent of any other position.

$$g_{ia} := \text{the probability of guessing the correct secondary structure at position i;} g_{ia} = \frac{1}{4}$$

$$g_i = 1 \text{ if } g_i = c_i \text{ else } 0$$

$$E[G] = E[\sum_{i=0}^{n-1}(1)(\frac{1}{4})] = \sum_{i=0}^{n-1}E[g_i] = \frac{n}{4}\text{; where n := the length of the amino acid sequence}$$

$$\text{G's accuracy} = \frac{\frac{n}{4}}{n} = \frac{1}{4}$$

As shown in **Figure 2**, our model performs above the threshold set by the random guessing baseline. In our experiments, we believe this is attributed to model bias and high data variance. Furthermore, since L can be highly non-convex - multiple similar conditional probabilities ratios can lead to the same objective - ordinary gradient descent can lead to minimums that generalize poorly to new amino acid sequences.

2.6 Leave-Out-One-Cross-Validation

To accurately estimate our validation error given a small dataset, we utilize a LOOCV (Leave-Out-One-Cross-Validation) to give a stronger estimation of our error. Leave-One-Out Cross-Validation (LOOCV) is an effective method for estimating the validation error, particularly when dealing with small datasets. Unlike other validation methods, LOOCV involves cycling through the dataset N times, where N is the number of data points in the dataset. In each cycle, one data point is left out for validation and the remaining N-1 points are used for training.

This process is repeated for each data point in the dataset, resulting in N separate training and validation sets. The key benefit of LOOCV over other methods, like k-fold cross-validation or a simple train/test split, is that it provides a more comprehensive assessment of the model's performance. Since every data point gets a chance to be in the validation set, LOOCV ensures that the model's performance is tested on all available data, reducing the risk of bias that can occur if certain patterns or outliers are only present in the training or validation sets.

Moreover, LOOCV is particularly useful in scenarios where the dataset is small. Small datasets pose the risk of high variance in the estimation of model performance. By using each data point for validation once, LOOCV helps in mitigating this risk, providing a more reliable and stable estimate of the model's performance. However, it's worth noting that LOOCV can be computationally expensive for large datasets, as it requires fitting the model N times. Despite this, for small datasets, the benefits of using LOOCV, in terms of obtaining a more accurate and less biased estimation of model performance, often outweigh the computational costs.

2.7 A Quick Detour on Error Decomposition

Expected test error can be decomposed into model variance, irreducible data noise, and model bias.

$$E_{x,y,D}[(h_D(x)-y)^2] = E_{x,D}[(h_D(x)-\bar{h}(x))^2] + E_{x,y}[(\bar{y}(x)-y)^2] + E_x[(\bar{h}(x)\bar{y}(x))^2]$$

Though, we cannot do anything about inherent data noise, our low model accuracy hints that our simplified model and small training set might have high bias and high variance.

2.8 Addressing Model Bias and Variance

In our naive implementation, we assign two hidden states, a hidden state representing high G+C content and another for low G+C content. Although this model is relatively simple to implement, it does not take long see that it struggles to capture complex relationships between amino acid sequences and their corresponding secondary structure, which directly affects its ability to generalize and predict secondary structures for new amino acid sequences (shown in the above figure).

Though not fully explored in this paper, we propose several possible models that will help to improve the results of identifying secondary structures.

Define four hidden states - each corresponding to a secondary structure that we define earlier. In this model, we now have 16 transition probabilities, 20 observed symbol emission probabilities for each state (80 total), and 4 class label probabilities for each state (16 total).

Define a hidden state for every possible pairing of amino acids. If there are 20 amino acids, this corresponds to $\binom{20}{2} = 190$ pairings. In this model, we have 190^2 transition probabilities, 20 observed symbol emission probabilities for each state (3800 total), and 4 class label probabilities for each state (760 total).

In our original training set, we only had 40 data points. So, we gathered large dataset from Uniprok, ultimately training our model on around 500 protein sequences for humans and 500 protein sequences for animals.

2.9 Stochastic Methods

To tackle non-convexity, we use stochastic gradient descent to increase our probability of escaping low-quality minimums, while also improving runtime complexity [5].

First, we show that in expectation, a noisy gradient is an unbiased estimate of the true gradient.

$$E[\Delta L(s_i, c_i|\theta)] = \frac{1}{n_i}\Delta L(s_i, c_i|\theta) = \Delta\left[\frac{1}{n_i}L(s_i, c_i|\theta)\right] = \Delta L(\theta)$$

However, although a noisy gradient reaches the same objective as the true gradient in expectation, variance can allow us to sidestep or completely overshoot minimums that are sharp in depth (risk of overfitting) or widely shallow (risk of poor generalization). We can reduce variance by increasing the number m of noisy gradients and taking their average, see **Algorithm 2**.

2.10 Adaptive Moment Estimation

To make our parameter tuning more adaptive, we perform an optimized variation of gradient descent (i.e Adaptive Moment Estimation). The algorithm uses exponentially weighted averages of the first and second moments of the gradient to update the model's parameters, see **Algorithm 3**.

Although Adaptive Moment Estimation fine tunes parameters by finding individual learning rates for each parameter, there are a few possible downsides to using this algorithm. The performance of Adaptive Moment Estimation tends to improve as the number of parameters increases but for models with a small number of parameters, the algorithm may struggle and find sub-optimal values instead. And similarly, we calculate class label sequence using Viterbi algorithm.

3 Results

As previously mentioned, we used real data for all of our protein sequences. We ran three models on the same set of human and rat protein sequences.

First, we will discuss the results of the gradient descent model as seen in **Figure 3**. The run time for this algorithm was significantly long which suggests the model may have had some difficulty converging. Nevertheless, we found that the model was able to predict human and rat sequences with 42% and 43% accuracy, respectively. In bother cases, the model performed better than random guessing.

We will now discuss the results of the modified Stochastic gradient descent model as seen in **Figure 4**. The run time for this algorithm significantly improved from the standard gradient descent as we were now only training out data on a batch of samples. The performance of this model was better than standard gradient descent as the accuracy for human and rat sequences was 56% AND 63%, respectively. It is notable that the model was able to predict protein structures of rat amino acid sequences. This suggests that there might be more correlation between protein structures and amino acid sequences in rats compared to humans or that rats have protein structures that are more common.

Finally, we will discuss the result of the modified Adaptive Moment Estimation gradient descent model as seen in **Figure 5**. The run time of this model was slightly faster than gradient descent and Stochastic gradient descent. In terms of performance, this model out-performed gradient descent as it's accuracy's for human and rat sequences was 0.53 and 0.50, respectively. However, it did not out-perform stochastic gradient descent, possibly because the model is less competent in generalizing data.

Overall, all three models performed about the same when predicting human protein sequences. On the other hand, Stochastic gradient descent had better accuracy when predicting rat protein sequences.

4 Discussion

Class Hidden Markov models allow us to incorporate class label sequence information in a supervised learning context, as well as give slight improvement in identifying amino acid sequences if there is indeed a strong relationship between secondary structures and the amino acid sequence itself. This can be done by numerical optimization methods (i.e. gradient descent - and its variations Stochastic gradient descent and Adaptive moment estimation). Furthermore, in situations where class emission probabilities are unknown, maximum likelihood estimation is a possible approach in estimating these quantities. We explored the effectiveness of the three mentioned optimization methods by training them on human and rat protein sequences.

While exploring the different estimation methods, we found that both Adaptive moment estimation and Stochastic gradient descent methods performed better than the standard gradient descent method as shown in **Figure 6**. Out of the three models, we found that Stochastic gradient descent performed the best on both human and rat protein sequences. However, in terms of run time, Adaptive moment estimation was the fastest model to converge.

One potential limitation is the lack of data. Some amino acids are less common than others and the model depends on data it's trained one. It is possible that we did not use enough sequences with the less common amino acids to train the model, thus leading to poor performance when we tested it on sequences with more rare amino acids. One possible solution that we could attempt in the future is to implement Laplacian smoothing. Another way to approach this issue is to strengthen the model's ability to process new sequences with some sort of regularization method. Another limitation we encountered is the time it took to run all three models. We required a significant amount of data split up into training our models and then, validating them. The run time for each model was very costly and took several hours to run trials.

Right now, CHMM incorporates an observed sequence and associated class labeling and makes no other assumptions about the data itself. Right now, there has been a great deal of sequential modeling and their relevance to natural language. It would be interesting to investigate the advantages and trade offs of incorporating positional embedding similar to an self-attention matrix in transformer models in regards to accuracy and bias.

Author Contributions

This section is required to all project groups with more than a single member. Place group member's name under the appropriate contribution.

- Study design: Alex, Andrew
- Coding: Alex, Andrew, Mason, Kaitlyn
- Experiments: Alex, Andrew, Mason, Kaitlyn
- Analyses: Alex, Andrew, Mason, Kaitlyn
- Writing:
  - *Introduction:* Alex, Mason, Jan
  - *Methods:* Alex, Andrew, Kaitlyn, Jan
  - *Results:* Alex, Kaitlyn, Jan
  - *Discussion:* Alex, Andrew, Mason, Kaitlyn, Jan

References

1. Krogh AS, 1994. Hidden markov models for labeled sequences. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition: Conference B: Computer Vision & Image Processing*, volume 2, pages 140–144. IEEE. Accessed 27 November 2023.

2. Krogh A, Brown M, Mian IS, Sjolander K, Haussler D, 1994. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531.

3. Krogh A, Mian IS, Haussler D, 1994. A hidden markov model that finds genes in e. coli dna. Technical Report UCSC-CRL-93-33, Computer and Information Sciences, Univ. of California at Santa Cruz.

4. Chan CY, et al., 2009. A structural interpretation of the effect of gc-content on efficiency of rna interference. *BMC Bioinformatics*, 10(S1):S33. Accessed 15 December 2023.

5. Ruder S, 2017. An overview of gradient descent optimization algorithms. arXiv.org. Accessed 27 November 2023.

Figures

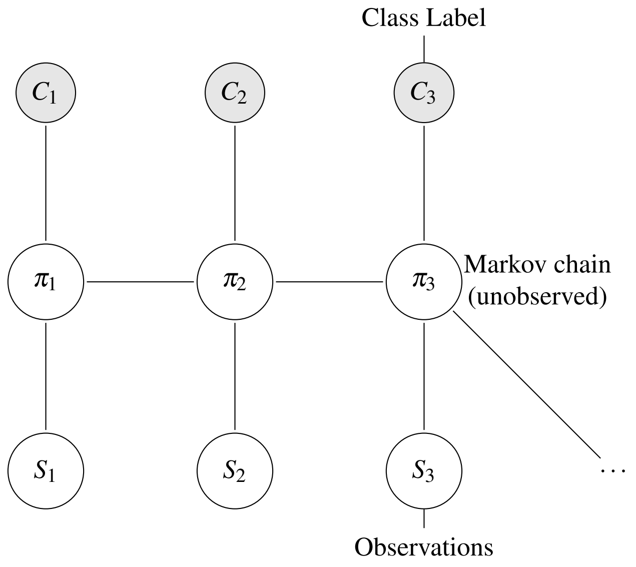

Figure 1. Class Hidden Markov Model.

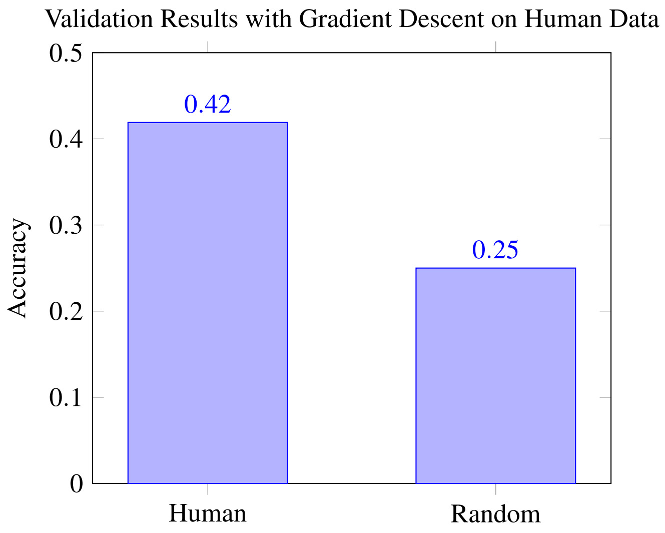

Figure 2. Comparison of CHMM performance on Human data with random guessing.

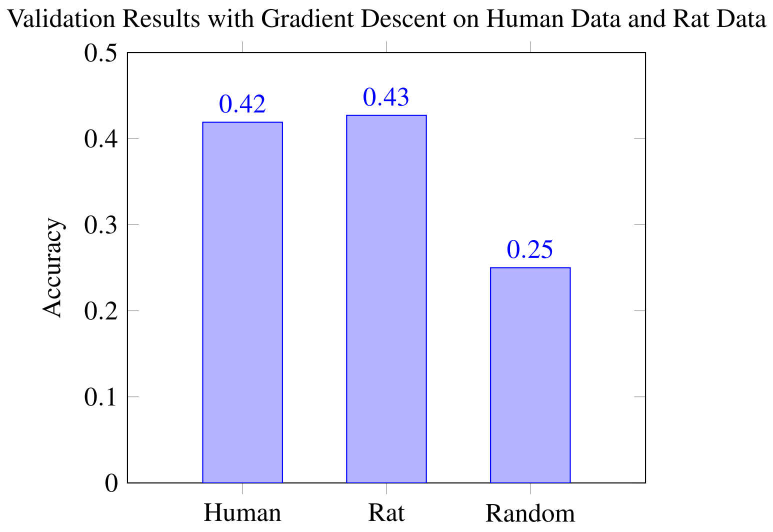

Figure 3. Comparison of CHMM performance on Human data and Rat data with random guessing.

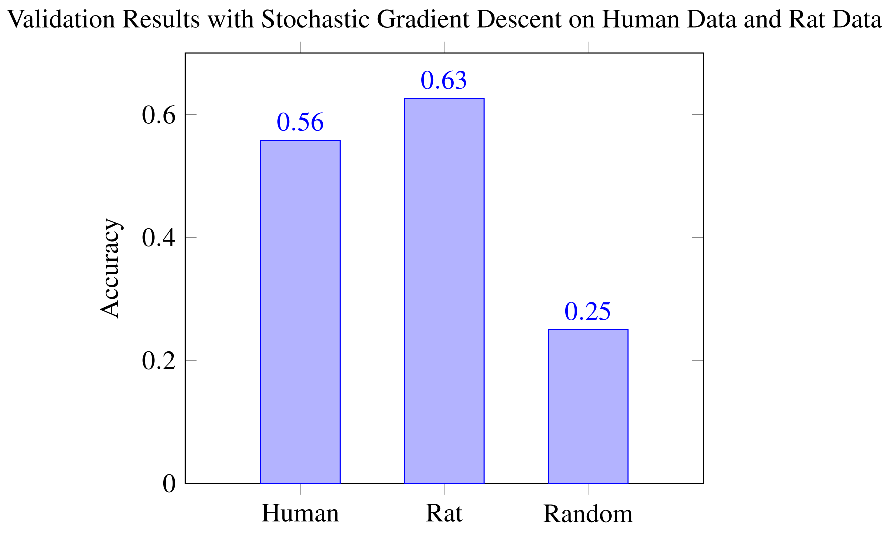

Figure 4. Comparison of CHMM performance on Human data and Rat data with random guessing.

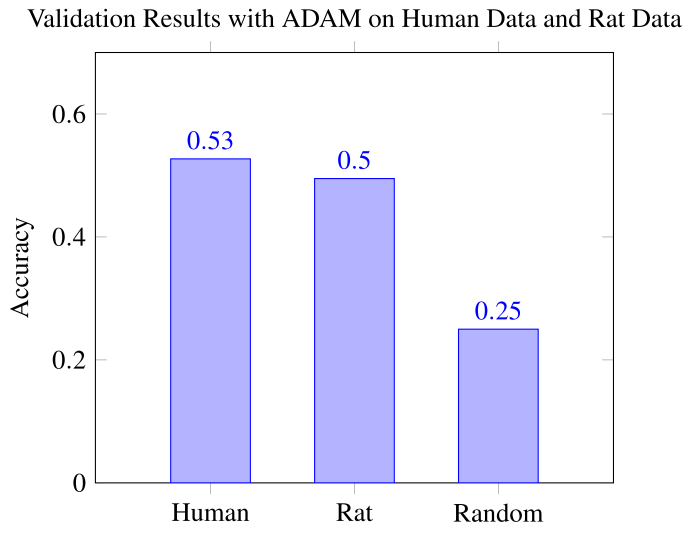

Figure 5. Comparison of CHMM performance on Human data and Rat data with random guessing.

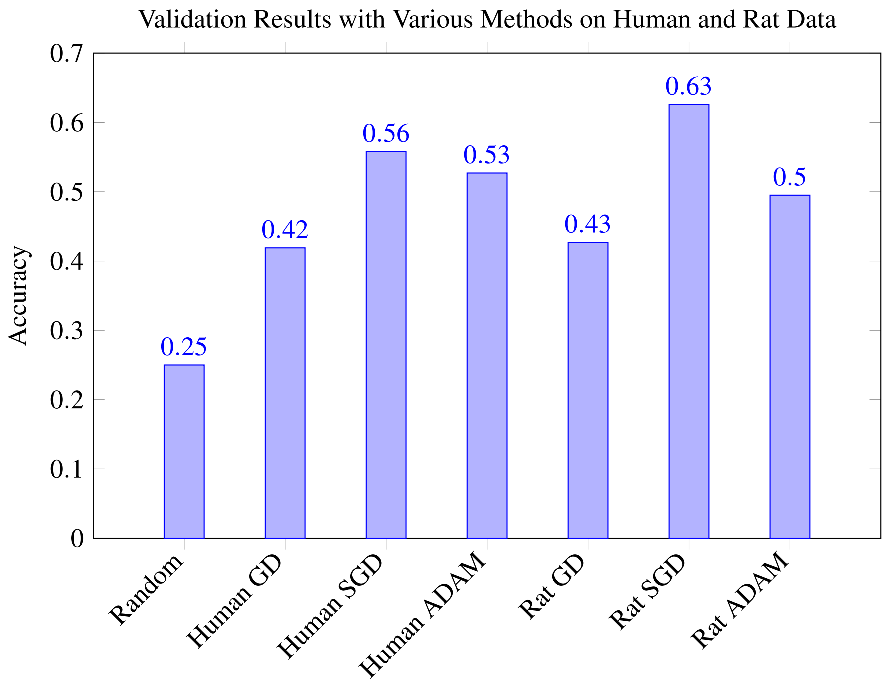

Figure 6. Comparison of CHMM performance on Human and Rat data using different optimization methods including Gradient Descent (GD), Stochastic Gradient Descent (SGD), and ADAM. Random guessing is used as a baseline.

Algorithms

1. Gradient Descent Algorithm

The stochastic gradient descent algorithm is described as follows:

I. While not converged:

- For each pair (s, c):
  - Calculate $n_k(s)$ using the forward-backward algorithm on all paths.
  - Calculate $m_k(c, s)$ using the forward-backward algorithm on permissible paths.
  - Update the gradient: $g_i \leftarrow -\frac{m_k(\mathbf{c},\mathbf{s}) - n_k(\mathbf{s})}{\theta_k}$.
- Compute the average gradient: $g \leftarrow \frac{1}{m} \sum_i g_i$.
- Update the parameters: $\theta \leftarrow \theta - \alpha g$.

2. Stochastic Gradient Descent Algorithm

The stochastic gradient descent algorithm for our model is as follows:

While not converged

I. Randomly shuffle amino acid sequence - class labeling pairs.

II. Randomly sample m pairs.

III. For each pair:

- Calculate $n_k(\mathbf{s})$ using the forward-backward algorithm on all paths.
- Calculate $m_k(\mathbf{c}, \mathbf{s})$ using the forward-backward algorithm on permissible paths.
- Update the gradient: $g_i \leftarrow -\frac{m_k(\mathbf{c},\mathbf{s}) - n_k(\mathbf{s})}{\theta_k}$.

IV. Compute the average gradient: $g \leftarrow \frac{1}{m} \sum_i g_i$.

V. Update the parameters: $\theta \leftarrow \theta - \alpha g$.

3. Adaptive Moment Estimation

The adaptive moment estimation algorithm for our model is as follows:

While not converged

I. Initialize moving averages

II. Compute moving averages at current iteration.

$$m_t = \beta_1 \cdot m_t + (1 - \beta_1) \cdot \left(\frac{\delta L}{\delta w_t} \right)$$

$$v_t = \beta_2 \cdot v_t + (1 - \beta_2) \cdot \left(\frac{\delta L}{\delta w_t} \right)^2$$

III. Bias correction

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

IV. Update parameters

$$w_t = w(t-1) - \alpha * (\hat{m}_t / \sqrt{(}\hat{v}_t) + e)$$