Zhenyu Wu
Prof. Brian Chen
CSE-308
17th April 2023

a) The main aim of the training phase is to improve the HMM model parameters, enabling it to effectively predict gene features within a particular DNA sequence. To do this, the training phase requires a dataset of DNA sequences with known genes and annotated gene features, such as coding and non-coding regions, start and stop codons, as well as the selection of the number and type of states and transition probabilities in the HMM structure. Initially, the model parameters are usually randomly estimated or derived from past knowledge.

The optimized set of HMM parameters generated from the training phase are then used to accurately model gene structure and composition in the input DNA sequences through the Expectation-Maximization (EM) algorithm. Metrics like sensitivity, specificity, accuracy, and F1 score are employed to measure the performance of the HMM during training, giving an indication of how well the model fits the training data and guiding the optimization process. Ultimately, the training phase produces a fully trained HMM model that can be utilized for gene prediction in new DNA sequences.

In the prediction phase, the trained HMM model is applied to a new DNA sequence to predict gene features. The input to this phase includes the DNA sequence to be analyzed and the trained HMM model from the training phase.

The main output of the prediction phase is a predicted set of gene features for the input DNA sequence, including the location and boundaries of coding and non-coding regions, start and stop codons, and the probability or confidence level of each predicted feature. The confidence score represents the likelihood of a predicted feature being a true gene based on the trained HMM model. These scores can be used to filter out low-confidence predictions and focus on the most likely gene features. Overall, the training and prediction phases of the HMM algorithm for gene prediction entail inputs and outputs designed to accurately model the patterns of gene structure and composition in a given DNA sequence.

b)In Markov Models, the graph structure represents the possible states and transitions between them.The graph structure is like a roadmap that outlines the potential states and transitions between them. It's kind of like a game of "connect the dots," where the dots are the states and the lines between them represent the transitions. The graph structure of a Markov Model is typically generated by defining the states and transitions based on prior knowledge of the system being modeled. For example, in gene prediction, the initial states and transitions might be based on known features of gene structure, such as the presence of start and stop codons. These initial states and transitions can then be refined and optimized using statistical methods such as the Expectation-Maximization algorithm.

The general graph structure of Markov Model is typically represented in memory as a set of nodes and edges. Each node represents a state in the model, and each edge represents a transition between two states. The nodes and edges are often stored in a matrix or list format, where each row and column of the matrix represents a state in the model and each entry in the matrix represents the probability of transitioning from one state to another. However, the memory representation can be varied based on the type of Markov Models. For instance, a Markov chain is the simplest type of Markov model, where all states are observable and probabilities converge over time. On the other hand, Hidden Markov Models are similar to

Markov chains, but they have a few hidden states that can't be observed directly in the chain, only through the observation of another process that depends on it.

Once the graph structure of Markov Model has been defined, the model parameters can be estimated using training data. This involves adjusting the probabilities associated with each node and edge in the graph to maximize the likelihood of the observed data. The resulting trained Markov Model can then be used to predict the most likely sequence of states for a new set of observations in a long term. In addition, the prediction can also be influenced by the type of Markov Model. For instance, compared to other types of Markov Model, a Markov chain has short-term memory, it only remembers where you are now and where you want to go next, which means the path you took to reach a particular state doesn't impact the likelihood of moving to another state. The only thing that can influence the likelihood of going from one state to the other is the state you are currently in.

c) Expectation Maximization (EM) algorithms are an essential tool for statistical inference when dealing with incomplete or hidden data. These algorithms are commonly used to estimate the parameters of a model, including initial state probabilities, emission probabilities, and transition probabilities between states. In the context of gene prediction, for instance, the observed data are the DNA sequences and annotated gene features, while the hidden or missing variables are the actual state sequence of the HMM that generated the observed data.

In essence, EM algorithms provide an approach for estimating maximum likelihood in the presence of latent variables. They find applications in various fields, such as clustering algorithms, factor analysis, and gene prediction.The EM algorithm works by iteratively updating estimates of the hidden or latent variables. It proceeds in two general steps: the expectation step (E-step) and the maximization step (M-step). In the E-step, the algorithm calculates the expected value of the latent variables, given the current estimates of the model parameters. This step typically uses Bayes' rule and the existing model parameter estimates to compute the expected value, also known as the posterior probability. In the M-step, the algorithm updates the model parameter estimates based on the expected values calculated in the E-step. It does this by finding the parameter values that maximize the log-likelihood of the observed data, given the expected values of the hidden or latent variables. The algorithm iteratively alternates between the E-step and M-step until it converges to the maximum likelihood estimate of the model parameters.

In addition, the specific steps of Expectation-Maximization (EM) algorithm for training Hidden Markov Models (HMMs) consists of four main steps: initialization, expectation, maximization, and checking of convergence. In the initialization step, the initial parameter values are set, and the incomplete observed data is fed into the system with the assumption that the data follows a specific probability distribution model. In the expectation step, the algorithm estimates or guesses the values of the missing or incomplete data using the observed data. This step updates the variables in the model. In the maximization step, the complete data generated in the expectation step is used to update the parameter values of the model. This step involves updating the hypothesis. The final step is the checking of convergence, where the algorithm checks whether the parameter values have converged to their final values or not. If convergence has not occurred, the expectation and maximization steps are repeated until convergence is achieved.

d) During the E-M algorithm, the edge weights are updated in the Maximization step using the expected values of the sufficient statistics that were computed in the Expectation step. To calculate the updated edge weights, the Maximization step typically involves normalizing the expected sufficient statistics over all possible transitions from a given state. Specifically, the expected count of transitioning from state i to state j is divided by the sum of the expected counts of all transitions from state i. This normalization

ensures that the sum of probabilities of all possible transitions from each state equals 1. The expected counts of the sufficient statistics are computed in the Expectation step of the E-M algorithm, where the probability of each state at each position in the observed data sequence is estimated using the current model parameters. These state probabilities are then used to estimate the expected counts of the sufficient statistics for each transition between states.

e)In a Hidden Markov Model (HMM) for DNA sequence prediction, each letter in the input sequence represents an observation that is emitted by the corresponding state in the model. The HMM consists of a set of hidden states, each of which emits a letter of the DNA alphabet with a certain probability. The model transitions between these states with certain probabilities as well, forming a probabilistic model of the underlying process generating the observed DNA sequence.

In more detail, the HMM for DNA sequence prediction typically involves a set of hidden states, where each state represents a different biological process or feature that may be occurring at a particular location in the DNA sequence. For example, there may be states representing exonic regions, intronic regions, regulatory regions, repetitive sequences. Each hidden state has an associated probability distribution over the DNA alphabet, which defines the probability of observing each letter of the alphabet given that the HMM is in that state.

The observed DNA sequence is then modeled as a sequence of emissions from these hidden states. Specifically, at each position in the DNA sequence, the HMM emits a letter of the alphabet with a probability given by the emission probability distribution of the current state. The model transitions between these states with certain probabilities as well, forming a probabilistic model of the underlying process generating the observed DNA sequence.

To make a prediction on the input DNA sequence, the trained HMM is used to compute the most likely state sequence that generated the observed sequence, using the Viterbi algorithm. The Viterbi algorithm can be used to find the maximum probability path through the model, given the observed sequence, and returns the corresponding state sequence. This state sequence can be used to make predictions about properties of the DNA sequence, such as regions that are more likely to be functional, conserved, or have certain structural features.

f)Determining the appropriate amount of data for training an HMM is a crucial task to prevent both overfitting and underfitting. Overfitting occurs when the model is too complex and is fitted too closely to the training data, leading to poor generalization and high error rates on new data. Underfitting, on the other hand, occurs when the model is too simple and cannot capture the patterns in the data, leading to poor performance on both the training and test datasets.

The amount of data required for HMM training depends on several factors, including the complexity of the model and the variability of the data. In general, larger datasets tend to produce better models with higher accuracy and generalization performance. However, it is also important to ensure that the data is representative of the true underlying distribution of the target population.

A commonly used approach to prevent overfitting is to use a separate validation dataset to evaluate the performance of the model during training, involving splitting the data into three sets: a training set, a validation set, and a test set. The model is trained on the training set, and the performance on the validation set is monitored. If the model begins to overfit to the training set, the performance on the validation set will begin to degrade, and the training can be stopped to prevent further overfitting. Another

approach of Statistical Heuristic is to use a technique called the learning curve, which plots the performance of the model as a function of the size of the training set. By analyzing the learning curve, one can determine the point at which the performance of the model plateaus or starts to degrade, which indicates that additional data is unlikely to improve the model's performance.

In general, the size of the training dataset should be large enough to provide a representative sample of the underlying distribution while also being manageable in terms of computational resources and time. The optimal dataset size may also depend on the specific application and the desired level of performance. In practice, a commonly used rule of thumb is to use at least 10 times as many data points as the number of parameters being estimated in the HMM model.

g) The result of an HMM algorithm in solving a specific problem depends heavily on the quality and appropriateness of the training data used. In the case of gene prediction, the training data should contain DNA sequences that contain known gene features, such as the locations of exons and introns. This is important because the HMM algorithm relies on identifying patterns and features within the training data to create accurate models that can predict the locations of genes in new sequences. Therefore, the training data should be representative of the problem being solved.

It is also important that the training data be diverse enough to capture the variability of gene features across different organisms and genomic regions. This is because the HMM algorithm should be able to handle variations and differences that can arise in different organisms or genomic regions. This can help to ensure that the HMM algorithm is accurate and can generalize well to new, unseen data.

The quality of the training data is also crucial for the performance of the HMM. The training data should be of high quality, with accurate annotations and minimal errors. This ensures that the HMM model is accurate and reliable, and that it can produce high-quality predictions. If the training data is of poor quality, the HMM model may be inaccurate and unreliable, leading to poor predictions.

Therefore one source of training data for HMM gene prediction is the set of annotated gene sequences in a reference genome. This data is often publicly available through databases such as Ensembl, a bioinformatics project to organize biological information around the sequences of large genomes which is a comprehensive source of stable automatic annotation of individual genomes,and can be downloaded in various file formats. Ensembl is a popular source for training HMMs for gene prediction because it provides a large and diverse collection of annotated genomes from various organisms. The genome annotations in Ensembl are obtained through a combination of experimental evidence and computational predictions, making them reliable and accurate. Ensembl also provides a standardized format for genome annotations, which facilitates the use of the data in HMM training. Moreover, Ensembl provides a comprehensive set of tools and resources for accessing and analyzing genomic data, making it a convenient source for researchers to obtain the necessary training data for their HMMs. The data in Ensembl are regularly updated and curated, ensuring that the training data remains relevant and up-to-date. Another source of training data is experimentally verified gene sequences that have been curated and annotated by domain experts. These datasets may be smaller and more focused on specific genes or organisms, but can provide high-quality training data with fewer errors or discrepancies.

h) In order to receive better results from HMM after training, the training data must be carefully selected and processed. When we receive the raw data, the first step is to perform data cleaning, a critical step in the data preparation process. By definition , data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. Just like the situation we

faced in project 1 where we might have thousands of duplicate paths after we perform the Eulerian Path Assembly, it will be a great possibility that we may receive numerous duplicate data in raw dataset which cause the program to go through the same data for many times and waste plenty of time and resources. Thus, cleaning of duplicates is the first step of data cleaning. In order to guarantee the accuracy of the result, cleaning of incorrect data is critical. Therefore, we also need to perform quality control, which means we need to remove data with relatively low quality, which is commonly seen in most data science and machine learning projects. Firstly, removal of the low-quality bases is required, which are individual nucleotide positions within a DNA sequence that have been assigned a low quality score by the sequencing machine. Those error-causing nucleotides position such as nucleotide positions that may have a higher likelihood of being read incorrectly due to their sequence context can be caused by a variety of factors, such as errors in the sequencing chemistry, poor quality of the starting DNA material, or issues with the sequencing instrument itself. Therefore, it is important to ensure that the error occurring in the sequencing stage will not affect the accuracy of the Markov Model.

Before we proceed, identifying the genomic location of each read and determining which genes or regions of the genome are expressed in the raw data is also critical. Therefore, sequence alignment might be required. In this case, we might need to perform an alignment algorithm, for example, Bowtie, a memory-efficient alignment of short DNA sequences to the human genome, to align reads to reference the genome to determine its location and origin. In addition, sequence alignment also allows us to identify variations and differences between the reference sequence and the sample being sequenced, which means we can quickly observe the variations by comparing raw sequences and their reference genome. Lastly, this step facilitates the data cleaning as well by removing reads that do not align well to the reference sequence, which can be caused by sequencing errors or contamination.

The next steps are data normalization and data transformation. Data normalization is the process of transforming raw data values into another format with properties better suited for modeling and analysis. In the article "*Maintenance Prediction through Sensing Using Hidden Markov Models—A Case Study*", researchers mentioned one type of data normalization which was called Z-score normalization. In the *Data Preparing* Session, they described Z-score normalization as a process to convert a variable range with some mathematical heuristics, allowing all variables to have the same range which could make patterns in the data more visible. Finally, normalized values may be transformed to fit the assumptions of the HMM model being used which is the data transformation step.

i)There are multiple reasons that can cause issues in the training process. The first and most obvious reason is insufficient training data. Just like I mentioned in the earlier session that the size of the training dataset should be large enough to provide a representative sample of the underlying distribution. If the training dataset is insufficient or not representative of the overall data distribution, the resulting model may not generalize well enough to new, unseen data.

In addition, the quality of the data set can also affect the training stage. Just like the data cleaning I mentioned in the previous section, it is possible that we might have data with relatively low quality and nucleotide positions that may have a higher likelihood of being read incorrectly. If the data cleaning and quality control steps are not performed well enough, those errors carried by the raw input data will continue to contaminate the training stage, producing problematic results.

The next reason is overfitting and underfitting. Underfitting happens when the model fits exactly against its training data which means it can not produce accurate results when againsting unseen data. This is a scenario that is commonly seen when the model is too simple with fewer parameters, like linear

regression. When underfitting occurs, the model cannot establish the dominant trend within the data, resulting in training errors and poor performance of the model, which can be identified by high bias and low variance. On the other hand, overfitting happens when a statistical model fits exactly against its training data, and is commonly seen in the complex model with more parameters. When a complex model is trained for too long on sample data, the model may start to learn some irrelevant information within the dataset include sequencing errors, artifacts, or biases introduced during the sequencing process, as well as non-coding regions of the genome that do not contain relevant biological information for the specific analysis being performed. For example, if the HMM is being trained to predict functional regions of the genome, such as coding exons or regulatory elements, non-coding regions such as introns or intergenic regions would be considered irrelevant information. In this situation, when the model memorizes the irrelevant information and fits too closely to the training set, the model becomes overfitted, and it is unable to generalize well enough to new data as well, which can be indicated through low error rates and a high variance. For the reason that the Markov model has multiple variations, such as the Markov chain, either the overfitting or underfitting will have certain possibilities to happen in the training stage.

j) There are several ways to accelerate the training process or perform training on a small dataset. The first method is transfer learning, a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. This can be achieved by pre-training the model on a larger or more diverse dataset, and fine-tuning it on the smaller training set. This can help the model to learn general features of the DNA sequence that are applicable across different datasets, and to adapt to the specific characteristics of the smaller training set. For example, one could pre-train an HMM on a large dataset of mammalian genomes, and then fine-tune the model on a smaller dataset of human genomes to predict specific features of the human genome.

Another approach is to use a form of unsupervised learning, which is a technique that applies a certain algorithm to analyze and cluster unlabeled datasets. Algorithms being used are those that can discover hidden patterns, similarities and differences in information, such as the Expectation-Maximization (EM) algorithm, which can estimate the parameters of the model from the observed data alone, without the need for labeled training examples. These algorithms iteratively estimate the parameters of the HMM by computing the expected probabilities of the hidden states given the observed data, and then using these probabilities to update the parameters. The process continues until convergence, at which point the HMM has learned the most likely set of parameters given the observed data.

The last approach is through regulation. Regularization applies a “penalty” to the input parameters with the larger coefficients, which subsequently limits the amount of variance in the model and encourages the model to have simpler parameter values, such as L1 regularization, Lasso regularization, and dropout. Through regulation, overfitting caused by insufficient data from a smaller dataset can be prevented, which helps us to choose the model with better complexity for the given dataset size and generalize better to new, unseen data.

Reference

Eddy, S. R. (1996). Hidden Markov models. Current Opinion in Structural Biology, 6(3), 361-365.

Brownlee, J. (2020, August 27). *A gentle introduction to expectation-maximization (EM algorithm)*. MachineLearningMastery.com. Retrieved April 17, 2023, from https://machinelearningmastery.com/expectation-maximization-em-algorithm/

Bento, C. (2021, December 8). Markov models and Markov chains explained in real life: Probabilistic Workout routine. Medium. Retrieved April 17, 2023, from https://towardsdatascience.com/markov-models-and-markov-chains-explained-in-real-life-probabilistic-workout-routine-65e47b5c9a73

Goyal, C. (2022, August 5). Complete guide to expectation-maximization algorithm. Analytics Vidhya. Retrieved April 17, 2023, from https://www.analyticsvidhya.com/blog/2021/05/complete-guide-to-expectation-maximization-algorithm/

Kwok, R. (2019, December 13). Viterbi algorithm for prediction with HMM - part 3 of the HMM series. Medium. Retrieved April 19, 2023, from https://medium.com/analytics-vidhya/viterbi-algorithm-for-prediction-with-hmm-part-3-of-the-hmm-series-6466ce2f5dc6

Birney, E., Andrews, T. D., Bevan, P., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cuff, J., Curwen, V., Cutts, T., Down, T., Eyras, E., Fernandez-Suarez, X. M., Gane, P., Gibbins, B., Gilbert, J., Hammond, M., Hotz, H.-R., Iyer, V., … Clamp, M. (2004, May). An overview of Ensembl. Genome research. Retrieved April 19, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC479121/

Martins, A., Fonseca, I., Farinha, J. T., Reis, J., & Cardoso, A. J. M. (2021, August 21). Maintenance prediction through sensing using Hidden Markov models-A case study. MDPI. Retrieved April 22, 2023, from https://www.mdpi.com/2076-3417/11/16/7685

Guide to data cleaning: Definition, benefits, components, and how to clean your data. Tableau. (n.d.). Retrieved April 22, 2023, from https://www.tableau.com/learn/articles/what-is-data-cleaning#:~:text=tools%20and%20software-,What%20is%20data%20cleaning%3F,to%20be%20duplicated%20or%20mislabeled.

Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. L. (2009, March 4). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome - genome biology. BioMed Central. Retrieved April 22, 2023, from https://genomebiology.biomedcentral.com/articles/10.1186/gb-2009-10-3-r25

What is underfitting? IBM. (n.d.). Retrieved April 22, 2023, from https://www.ibm.com/topics/underfitting#:~:text=the%20next%20step-,What%20is%20underfitting%3F,training%20set%20and%20unseen%20data.

What is overfitting? IBM. (n.d.). Retrieved April 22, 2023, from https://www.ibm.com/topics/underfitting#:~:text=the%20next%20step-,What%20is%20underfitting%3F,training%20set%20and%20unseen%20data.

Priya_dharshini__. (2021, June 13). 4 ways to handle insufficient data in machine learning! Analytics Vidhya. Retrieved April 22, 2023, from https://www.analyticsvidhya.com/blog/2021/06/4-ways-to-handle-insufficient-data-in-machine-learning/

Brownlee, J. (2019, September 16). A gentle introduction to transfer learning for Deep learning. MachineLearningMastery.com. Retrieved April 22, 2023, from https://machinelearningmastery.com/transfer-learning-for-deep-learning/

What is unsupervised learning? IBM. (n.d.). Retrieved April 22, 2023, from https://www.ibm.com/topics/unsupervised-learning#:~:text=the%20next%20step-,What%20is%20unsupervised%20learning%3F,the%20need%20for%20human%20intervention.