A Novel Algorithm for Training Hidden Markov Models with Positive and Negative Examples

Jiefu Li
Computer and Information Sciences
University of Delaware
Newark, United States
lijiefu@udel.edu

Jung-Youn Lee
Plant and Soil Sciences
University of Delaware
Newark, United States
jylee@udel.edu

Li Liao(✉)
Computer and Information Sciences
University of Delaware
Newark, United States
liliao@udel.edu

***Abstract*—In this paper, we present a novel training method based on Baum-Welch algorithm for hidden Markov models (HMM), named as Comprehensive HMM (CompHMM), which changes the traditional approach of training HMM from positive examples only to be able to utilize both positive and negative examples in training HMMs. By comparison, our method outperformed the standard Baum-Welch method and another HMM discriminative training method significantly through both synthetic and real data in membership prediction task.**

***Index Terms*—HMM, negative examples**

I. Introduction

Hidden Markov models (HMM) [1]–[5] are a well known statistical tool for modelling sequential data. HMMs have been successfully applied to many problems in the fields of nature language processing and bioinformatics, such as speech recognition [6], biological sequence alignment [7], and CpG island prediction [8]. In applications, HMMs are commonly used for two different tasks. One task is membership prediction: to compute the probability of a given sequence belonging to a certain group, which is modelled by the HMM. The other task is called decoding: to compute the most probably hidden path for the given sequence. In the field of nature language processing, HMMs are mainly used for decoding. One example is speech recognition: translate human voice (observable sequence) to plain text (hidden path). In the field of bioinformatics, HMMs have been used for both membership prediction and decoding. One successful example is HMMER [7], which is a special case of HMM called profile HMM (pHMM). HMMER is used for two purposes: sequence alignment and remotely homologous detection. Sequence alignment task uses HMMER for decoding: HMMER computes the insertion, deletion, or mutation state (hidden path) for the given sequence (observable sequence) position wise. Remote homologous detection task uses HMMER for membership prediction: HMMER computes the likelihood score of a sequence belonging to certain protein family. Over the years, HMMs have had great successful applications in different domains, like most methods in machine learning, their successes rely to a large degree on both the quantity and quality of data. Meanwhile, there are still many situations where both quantity and quality of the data are poor, preventing any machine learning tools to fully capture the knowledge of the data, especially for ongoing biological research [9]. And the fact that HMMs use only positive examples for training exacerbates the issue of limited training data, by leaving out negative examples with no use. To tackle this issue, in this paper, we develop a novel Baum-Welch based training algorithm, which utilizes both positive and negative data to train HMMs in order to enhance the membership prediction task for biological sequences, when the number of positive examples is limited.

This work is inspired by our recent research on detecting novel Plasmodesmata-located proteins (PDLPs) [9]. PDLPs are type I transmembrane proteins, which are targeted to intercellular pores called plasmodesmata that form at the cellular junctions in plants [10]. Currently, only 8 PDLPs have been verified in Arabidopsis thaliana experimentally, and their PD-targeting signal regions have been narrowed down to extracellular juxtamembrane domain (JMe). In [9], a 3-state HMM called PDHMM was designed to model PDLPs' JMe region for detecting de novo PD-targeting signal and identifying novel PDLPs. By wet-lab experiments, several predicted PD-targeting signals have been successfully verified. However, identifying de novo PDLPs still remains a challenge: for those proteins ranked highly by PDHMM as PD-targeting candidates, some are found to be none-PD as reported in the literature and many others are still unknown. Due to the high cost of wet-lab experiments, an improved HMM for PDLP prediction has become even more imperative for this ongoing research. One way to improve the model is having more data, and due to the limited number of positive data, an HMM training algorithm of utilizing both positive and negative data is urgently needed.

In this paper, we proposed a novel training method to utilize both positive and negative data for training a "comprehensive" HMM, or CompHMM for short. Hereafter, without confusion, we use CompHMM to refer to both the training method and the model thus trained. CompHMM train the HMM in a discriminative fashion: the positive and negative examples are separated in its measure space. In contrast, conventional training algorithms for HMMs, such as Baum-Welch, Viterbi training, and maximum likelihood, are considered as a positive-only approach, because they use only

978-1-7281-6215-7/20/$31.00 ©2020 IEEE

positive data for training. As such, HMMs belong to a category of models, called generative models in the field of machine learning, which means that the trained model can be used to generate synthetic examples with varied likelihood of being a positive example, but cannot give a clear decision boundary, although in practice 50% likelihood may be used as a de facto threshold. On the other hand, classifiers such as support vector machines that give a clear decision boundary are often referred to as discriminative model. Note that, although trained in a discrinimative manner, the trained CompHMM is a hidden Markov model nonetheless, and hence does not give a clear decision boundary. While there have been efforts to remedy the positive-only training limitation of HMMs by combining it with another discriminative classifier such as SVM [11] [12], our goal here is to modify the training within HMMs so that both positive and negative examples can be used without resorting to another discriminative classifier. The most relevant work to ours was the one carried out Mamitsuka [13]. This method, named as "MA", also attempts to have a discriminative manner training for HMM but is based on gradient descent technique developed in [14]. One apparent advantage of CompHMM over MA method is that it has fewer hyper-parameters to select. In MA method, there are four hyper-parameters, and CompHMM only has one, which will be defined in section II-D. More importantly, based on experimental results from both synthetic and real data, our method outperformed standard Baum-Welch and MA method significantly.

The rest of this paper is organized as the following. In the Method section, a brief review of HMMs is first given, mainly to introduce the notations and revisit the standard Baum-Welch algorithm. What follows is a key step in our new method, called reverse Baum-Welch. Then the whole procedure of CompHMM is presented and explained. In the results section, we describe in details the generation of synthetic data and the setup of experiments for testing the methods with both synthetic and real data, and compare the performance of our CompHMM with that of standard Baum-Welch and MA methods.

II. Method

A. Introduction to HMM

HMMs are a well known tool in machine learning field for modeling sequential data, mainly to discover underlying structures and patterns. One of the concepts in modeling sequential data with HMMs is to define hidden states corresponding to those underlying structures/patterns, besides the data, which is called "observable". For example, given a genomic sequence (observable), we want to know whether a nucleotide is in a gene (hidden state) or an intergenic region (another hidden state). The core assumption of modeling data in such a way is that the observation is emitted from the hidden states by some probabilities and the a hidden state can transition to another hidden state with certain probability, and the path of state to state transitions forms a Markov chain. And as the name of HMM suggested, it is designed to capture this observable and hidden relationship of sequential data.

A hidden Markov model is referred to by its parameters, denoted as θ collectively, which contains three sets of parameters $\theta = (\pi, A, B)$. π is called initial probability over the N states of the model, from which the hidden Markov chain starts with, in other words, π's element π_i stands for the probability of being state i at time $t = 1$. A is called transition matrix with dimension of $N \times N$ whose elements denoted as $a_{i,j}$ gives the probability of transition from state i to state j. B is called emission matrix with dimension of $N \times K$ whose elements denoted as $b_j(k)$, which stands for the probability of being at state j and emitted observation (symbol) k, where K is the size of alphabets (or symbols) of the observable data.

With a HMM θ and a sequence of observation O whose elements are $O_t \in K$, where $t = 1..T$. There are three tasks in data modeling with HMMs. One task is to compute $Pr(O|\theta)$, namely, the probability that sequence O is a member of the sequence family described by HMM θ. This probability, also called likelihood when considered as a function of θ, is computed by the so-called Forward algorithm efficiently. The second task is to train the HMM so that for m sequences, $\sum_{s=1}^{m} Pr(O^s|\theta)$ is maximized. When the sequence of hidden states or hidden path is given, this is done simply by maximum likelihood approach, namely by counting the numbers of various transitions and emissions, followed by proper normalization. When only observation is available, one popular method for training is Baum-Welch algorithm. The third task is called decoding task. This task is to find the most probable hidden state sequence X^* for a given observation sequence O and model θ: $X^* = \text{argmax}_X \, Pr(O, X|\theta)$. This task is carried out using Viterbi algorithm and is not the focus of this paper. For readers who are interested in knowing more about HMM, a comprehensive introduction can be found in [15]. In this paper, the focus is on the second task as discussed above. In the second task, Baum-Welch algorthim trains the HMM (i.e., adjusting its parameters) by maximizing the total probability over a set of observed sequences whose hidden state path are not available. Let this particular set of sequences called positive set. The assumption is that these sequences share commonalities in a way that they can be collectively described by a hidden Markov model. As such, the training is to fix the model parameters θ to maximize the likelihood, which is the principle of maximum likelihood. When the data are unlabeled, namely, the hidden state paths are latent, the technique for maximizing the likelihood is called Expectation-Maximization, see next subsection for details. After the training, one can use the trained HMM to calculate the log-likelihood of any given sequence and use this log-likelihood as a score metric to determine how likely the sequence belongs to this positive set. The focus of this paper is to train the HMM discriminatively so that the model can separate the target positive set from some negative sets and gain additional differentiating power. In table I, all notations used throughout this paper are listed with explanations.

TABLE I
NOTATIONS

Symbols	Explanations
θ	Hidden Markov model: $\theta = (\pi, A, B)$
N	States' number in hidden Markov model.
K	Symbolic Number in hidden Markov model.
A	Transition matrix with dimension $N \times N$.
a_{ij}	Probability of state i transition to state j.
B	Emission matrix with dimension $N \times K$.
$b_j(k)$	Probability of symbol k emitted from state j.
π	Initial probability of states with dimension $N \times 1$.
O^s	The s^{th} sequence with length T^s
X^s	State sequence of O^s
m	Total number of sequences

B. Baum-Welch Revisit

Since CompHMM is built based on Baum-Welch algorithm, a revisit of it will be helpful as the first step, though only the parts necessary for our discussion are presented here. More details of Baum-Welch algorithm can be found in [15].

Baum-Welch algorithm is an expectation-maximization (EM) algorithm, which has two steps: expectation (E) and maximization (M) steps. The E-step's calculation needs two matrices with dimension of $N \times T$ given HMM's state number N, and a sequence with length T. These two matrices are called forward and backward matrices and denoted as α and β respectively. Forward matrix's element $\alpha_i(t)$ stands for the probability of the symbol O_t being emitted from state i at time t given the hidden Markov model θ and the sub-sequence before time t: $O_{q=1\ldots t}$. Backward matrix's element $\beta_i(t)$ stands for the probability of the symbol O_t being emitted from state i at time t given model θ and the sub-sequence after time t: $O_{q=t+1\ldots T}$. They are all calculated recursively by forward-backward procedure as follows.

Baum-Welch algorithm starts with initial guesses of transition and emission matrices A and B. Then for each state $i \in N$, the value of α at time $t = 1$ is initialized by π:

$$\alpha_i(1) = \pi b_i(O_1) \tag{1}$$

With the above initialization, the remaining values of α are calculated recursively:

$$\alpha_i(t+1) = b_i(O_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji} \tag{2}$$

Similar to α's recursion calculation, β's calculation is backward of the time and initialized with all ones at time $t = T$:

$$\beta_i(T) = 1 \tag{3}$$

Then, the remaining values of β are calculated backward recursively:

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(O(t+1)) \tag{4}$$

With α and β, we can calculate the posterior state probabilities γ whose element $\gamma_i(t)$ stands for the probability of the symbol O_t visit state i at time t given the full sequence O.

$$\gamma_i(t) = P(X(t) = i|\theta, O) = \frac{P(X(t) = i, O|\theta)}{P(O|\theta)} \tag{5}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \tag{6}$$

And also $\xi_{ij}(t)$, which stands for the summed probability of all possible hidden paths with the restrict that at time t the symbol O_t is emitted from state i and at time t+1 O_{t+1} is emitted from state j.

$$\xi_{ij}(t) = \frac{P(X(t) = i, X(t+1) = j, O|\theta)}{P(O|\theta)} \tag{7}$$

$$= \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(O(t+1))}{P(O|\theta)} \tag{8}$$

At this point, the expectation information is captured in γ and ξ. The next step is M-step: maximization.

In case of multiple sequences, the updated probability of state i transit to state j, which denoted as a^*_{ij}, and updated probability of observing symbol o_k given at state i, which denoted as $b^*_i(o_k)$ are calculated as the follows.

$$a^*_{ij} = \frac{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \xi^s_{ij}(t)}{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \gamma^s_i(t)} \tag{9}$$

$$b^*_i(o_k) = \frac{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \gamma^s_i(t) I_{O^s(t)=o_k}}{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \gamma^s_i(t)} \tag{10}$$

where $I_{O^s(t)=o_k}$ is 1 if $O^s(t) = o_k$, and is zero otherwise. Replace the elements of A and B by a^*_{ij}, $b^*_i(o_k)$ correspondingly. Repeat above steps ((1) to (10)) and updating of A and B until $\sum_{s=1}^{m} Pr(O^s|\theta)$ converges with certain threshold or reaches the maximum iterations set by the user.

C. Reverse Baum-Welch

The reverse task of training HMM can be described as finding HMM parameters θ^- so that $\sum_{s=1}^{m^-} Pr(O^{-s}|\theta^-)$ is minimized with m^- number of sequences denoted as O^{-s}. Our reverse Baum-Welch is not precisely defined for such a task as the goal, rather it is defined a procedure to incorporate negative examples in order to combine with the parameters trained by standard Baum-Welch algorithm so that the training can be "comprehensive", namely, taking into account of both positive and negative examples.

The E-step is identical to standard Baum-Welch: calculations of α, β, γ, and ξ. And we denoted them as α^-, β^-, γ^-, and ξ^- correspondingly in order to differentiate reverse Baum-Welch from the standard Baum-Welch. The difference is in M-step:

$$a^{-*}_{ij} = \frac{C - \sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \xi^{-s}_{ij}(t)}{\sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \gamma^{-s}_i(t)} \tag{11}$$

$$b^{-*}_i(o_k) = \frac{C - \sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \gamma^{-s}_i(t) I_{O^{-s}(t)=o_k}}{\sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \gamma^{-s}_i(t)} \tag{12}$$

Where C is a constant defined as $C = \sum_{s=1}^{m^-} Length(O^{-s})$. The constant C makes sure that a_{ij}^{-*} and $b_i^{-*}(o_k)$ are all positive values. Because of the usage of C, a_{ij}^{-*} and $b_i^{-*}(o_k)$ need to be normalized to stand for probabilities as the ones in (10):

$$\overline{a_{ij}^{-*}} = \frac{a_{ij}^{-*}}{\sum_{q=1}^{N} a_{iq}^{-*}} \tag{13}$$

$$\overline{b_i^{-*}(o_k)} = \frac{b_i^{-*}(o_k)}{\sum_{o_q=1}^{K} b_i^{-*}(o_q)} \tag{14}$$

Then the iterations part and stopping criterion are same as standard Baum-Welch algorithm.

The rationale of reverse Baum-Welch is simple. One way to interpret it is to think of the constant C as a uniform pseudo count for emissions and transitions, and the training effect from a negative example O^- for each iteration is to negatively "engrave" this background, namely subtracting C by the actual counts from O^-.This is in contrast to standard Baum-Welch algorithm, which can be considered as "embossing" on an flat empty background. Or one can understand reverse Buam-Welch as there are sequences O^* as opposite of O^- implicitly, with the initial guess of/ last updated model, sequences O^* lead to the updated $\overline{a_{ij}^{-*}}$ and $\overline{b_i^{-*}(o_k)}$ with standard Baum-Welch algorithm, and try to fit model parameters θ^- with O^*.

There are a few points we like to emphasize. First, it is not necessary to develop a reverse version of Baum-Welch algorithm to find θ^- to minimize $\sum_{s=1}^{m^-} Pr(O^{-s}|\theta^-)$; such minimization can be achieved using gradient descent technique [14]. Rather, the reverse Baum-Welch is a part of CompHMM, not meant to be used stand-alone for the aforementioned minimization. Second, the reverse Baum-Welch cannot guarantee a monotonous decrease of $\sum_{s=1}^{m^-} Pr(O^{-s}|\theta^-)$, the reason of this directly follows reverse Baum-Welch's second interpretation: the imaged sequences O^* are different for each iteration.

D. CompHMM

Equipped with the standard Baum-Welch and reverse Baum-Welch, it is now quite straightforward to describe CompHMM. First, let us define the task. Given m^+ number of positive sequences denoted as O^+, and m^- number of negative sequences denoted as O^-. The task is to find a HMM θ so that $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ is maximized. Such a HMM is CompHMM, since it is trained comprehensively, taking into account of both positive and negative examples.

With initial parameters θ and positive sequences O^+, obtain its updated transition and emission probabilities:

$$a_{ij}^{*+} = \frac{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \xi_{ij}^{+s}(t)}{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \gamma_i^{+s}(t)} \tag{15}$$

$$b_i^{*+}(o_k) = \frac{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \gamma_i^{+s}(t) I_{O^{+s}(t)=o_k}}{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \gamma_i^{+s}(t)} \tag{16}$$

Then, with θ and negative sequences O^-, using reverse Baum-Welch to obtain its updated transition and emission probabilities: a_{ij}^{-*} and $b_i^{-*}(o_k)$. Finally, with a hyper-parameter r, chosen by user in range of $(0, 1)$, is used as a weighting factor to combine "embossing" effect from positive examples and "emgraving" effect from negative examples. So the final updating equation is:

$$a_{ij}^{*} = \frac{r \times a_{ij}^{*+} + (1-r) \times a_{ij}^{*-}}{\sum_{j=1}^{N} r \times a_{ij}^{*+} + (1-r) \times a_{ij}^{*-}} \tag{17}$$

$$b_i^{*}(o_k) = \frac{r \times b_i^{*+}(o_k) + (1-r) \times b_i^{*-}(o_k)}{\sum_{o_k=1}^{K} r \times b_i^{*+}(o_k) + (1-r) \times b_i^{*-}(o_k)} \tag{18}$$

For each iteration, update θ according to (17) and (18). Repeat this procedures to update θ, until $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ converges with certain tolerance or reaches the maximum iteration set by users. Note that for membership prediction task, the scoring metric for HMM is the log-likelihood of $P(O|\theta)$, which decided majorly by emission matrix B, therefore, it is reasonable to just make $a_{ij}^{*} = a_{ij}^{*+}$. In our experiment section, this method is adopted.

Even though due to similar reason of reverse Baum-Welch, CompHMM cannot guarantee monotonous increase of $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ for each iteration, it happens in majority iterations. By experiments, only its few beginning iterations show decrease or oscillation behaviors.

III. Experiments and Results

In this section, we will first present a training validation check of CompHMMs in order to have an intuitive understanding of the CompHMM. Then, a series of experiments of both synthetic and real data will be performed.

A. Training validation check

The primary goal here is to set up experiment to monitor CompHMM's training phase in order to validate that training procedure indeed increases the separation of positive and negative examples in their measured space, or equivalently, $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ is maximized.

In the experiment, we randomly initialized two hidden Markov models with state number of 3, alphabetic number of 20. One of them served as positive model, the other served as negative model. Each of them generates 50 sequences, each of which has a length of 30. Together they form a dataset containing 50 positive sequences and 50 negative sequences. The initial guess of transition and emission matrices were generated randomly with same size as the positive/negative model. We trained the CompHMM with a maximum number of 1500 iterations, tolerance of convergence set to be $1e-6$, hyper-parameter $r = 0.5$. The experiment shows that the algorithm reaches convergence at 635^{th} iteration. The tracking of $log(P(O^+|\theta))$, $log(P(O^-|\theta))$ are shown in Fig. 1, and the tracking of $log(P(O^+|\theta)) - log(P(O^-|\theta))$ is shown in Fig. 2 as the following.

The above results indicate that CompHMM meets the designing purpose, which utilize both positive and negative data and separate them gradually iteration by iteration, measured by log-likelihood.

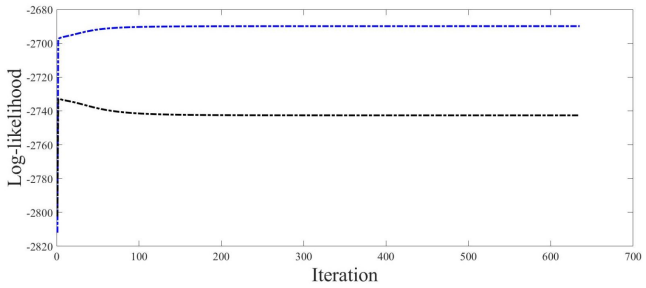

Fig. 1. Log-likelihood track of positive (blue) and negative (black) examples.

TABLE II
COMPARISONS FOR THE METHODS OF COMPHMM,MA, BW, GROUND TRUTH IN DIFFERENT SETTINGS

# of state / # of negative models	CompHMM	MA	BW	ground truth
3/5	0.7752	0.6683	0.7080	0.7987
3/10	0.7758	0.6551	0.7065	0.8003
5/5	0.7121	0.6012	0.6049	0.7455
5/10	0.7100	0.5963	0.6071	0.7446
7/5	0.6549	0.5601	0.5453	0.6922
7/10	0.6590	0.5649	0.5563	0.6990

B. Comparison study on synthetic data

After the training validation check of CompHMM, in this subsection, we designed a series of experiments with synthetic data to test CompHMM's performance on membership prediction task.

For each trial of experiments, the synthetic data generated by 1 positive model randomly generated, and $5/10$ negative models, which are generated from positive model by injecting random noise. We vary the injected noise level in order to control the difficulty levels of membership prediction, which are "calibrated" with the positive model being used as ground truth. That is, as the injected noise increases, we expect to see that the ground truth model has better performance in membership prediction, i.e., recognizing its own sequences out of the sequences generated by the negative models. The synthetic data contains 120 sequences in total, all with length of 30. 50% of these sequences are used as training and the others are used as testing. Among both training and testing sets, the numbers of positive and negative sequences are equal.

We also test training difficulty levels in terms of the model complexity. We chose state number to be 3, 5, and 7, all with alphabetic length of 20 to mimic amino acids in biological field. When state number is fixed, the state number of positive model, negative models, and learning models are the same. The prediction results are evaluated using AUC-ROC as evaluation metric [12]. The sequences in a test set is ranked by their log-likelihood score from the trained model, and a sliding threshold goes from the top to the bottom of the ranked list, one sequence on a time. At each time, the sequences at and above the threshold are predicted as positive and sequences below the threshold are predicted as negative. The predictions are checked with the ground truth label to be determined as true positive or false positive. The ROC curve is a plot of the true positive rate as a function of the false positive rate as the sliding threshold scans through the list. AUC-ROC is the area under the ROC curve, ranging from 0 to 1. A random classifier with no differentiating power would receive a AUC-ROC value of 0.5, and the higher the value the more differentiating power a classifier has, with AUC-ROC of 1.0 for a perfect classifier. In this study, each experiment is repeated for 30 times to mitigate random effects. Fig. 3 shows the performances of various methods from experiments with the synthetic data generated by state number of 5, and 10 negative models. As it can be seen, as the "easiness" increases – negative data easier to be differentiated from the positive data, the performance increases for the ground truth model, as expected. Across the whole range of "easiness", our method performs closely to the ground truth model and significantly outperforms the standard Baum-Welch and the MA method.

The complete results of the synthetic experiments are summarized in table II. Like Fig. 3, table II shows that: (1) CompHMM performs strictly better than both BW and MA, and closely to ground truth model; (2) MA performs actually worse than standard BW with the synthetic data when negative sequences come from an ensemble of many different negative models, a situation that is more closely mimic to the reality.

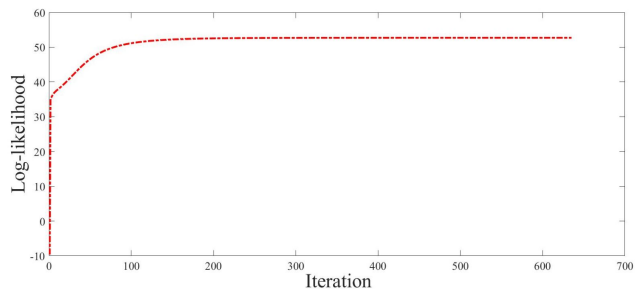

Fig. 2. Log-likelihood track of the difference between positive and negative examples.

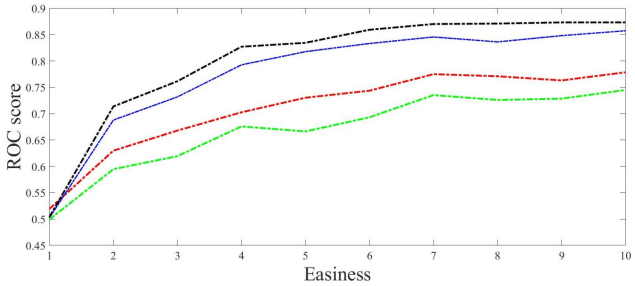

Fig. 3. AUC-ROC comparison for CompHMM (blue), BW (red), MA (green), and ground truth model (black) for state number = 5, with 10 negative models in different easiness level for membership prediction.

C. Applying on real data: plasmodesmata-associated protein membership prediction

To further test our method, we have applied CompHMM to real data. In our previous work [9], we used a simple 3-state HMM called PDHMM to model Plasmodesmata-located proteins (PDLP)' juxta membrane region, which contains the Plasmodesmata targeting signals, to perform PD vs none-PD classification. Later, 13 additional Plasmodesmata-associated proteins are noticed in literature. Plasmodesmata-associated proteins (PAP) are PD-targeting proteins but have different topology from PDLPs and are very likely from different gene families. Therefore PDHMM, trained on PDLPs, is expected to have lesser performance of predicting PAPs than predicting PDLPs. This can be a good case study of the improvements gained by applying CompHMM.

In this experiment, 8 PDLPs are used as positive sequences for training, and 13 additional Plasmodesmata-associated proteins are used as positive sequences for testing. There are also 365 other type I transmembrane proteins served as negative sequences for both training and testing. A 20 folds cross-validation performed, and each fold only splits the 365 other type I transmembrane proteins into negative training set and negative testing set. Positive training's 8 PDLPs and positive testing's 13 Plasmodesmata-associated proteins are used repeatedly for each fold.

The graphic result is shown in Fig. 4. The AUC-ROC of CompHMM, MA, and BW are 0.7125, 0.6254, 0.6005 respectively, which once again shows ComHMM's significant improvement over both MA and BW.

IV. Conclusion and Future Work

In conclusion, based on the standard Baum-Welch algorithm, we developed a novel method for training HMM that enables the use of both positive and negative examples to build a more reliable model. Experiments in membership prediction task with synthetic data and real data show that our method significantly outperformed both the standard Baum-Welch algorithm and the MA method. The latter is a gradient-descent based method to use both positive and negative examples to train HMMs.

While the improvements gained by utilizing negative examples shown in this study are remarkable, a few issues remain to be further investigated in the future. First, the effect of hyper-parameters r, which is used as a weighting factor between positive and negative examples, needs to be studied. Second, as mentioned in II-C, convergence of CompHMM cannot be guaranteed like standard Baum-Welch algorithm. Therefore, future study will focus on how to improve the algorithm so that the convergence is guaranteed or what can be done practically to mitigate the issue when its convergence cannot be guaranteed theoretically. Moreover, CompHMM will be deployed to our ongoing research of discovering de novo Plasmodesmata-located proteins.

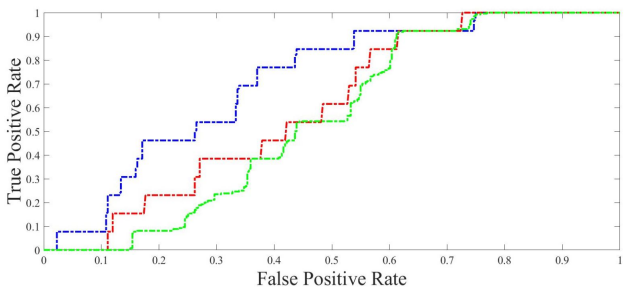

Fig. 4. ROC curves of CompHMM (blue), BW (red), MA (green) for Plasmodesmata-associated protein membership prediction.

Acknowledgment

The work is funded by National Science Foundation NSF-MCB1820103.

References

[1] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[2] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology," *Bulletin of the American Mathematical Society*, vol. 73, no. 3, pp. 360–363, 1967.

[3] L. E. Baum and G. Sell, "Growth transformations for functions on manifolds," *Pacific Journal of Mathematics*, vol. 27, no. 2, pp. 211–227, 1968.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[5] L. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process," *Inequalities*, vol. 3, pp. 1–8, 1972.

[6] L. Bahl, P. Brown, P. De Souza, and R. Mercer, "Maximum mutual information estimation of hidden markov model parameters for speech recognition," in *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 11. IEEE, 1986, pp. 49–52.

[7] R. D. Finn, J. Clements, and S. R. Eddy, "Hmmer web server: interactive sequence similarity searching," *Nucleic acids research*, vol. 39, no. suppl_2, pp. W29–W37, 2011.

[8] A. Barazandeh, M. Mohammadabadi, M. Ghaderi-Zefrehei, and H. Nezamabadipour, "Predicting cpg islands and their relationship with genomic feature in cattle by hidden markov model algorithm," *Iranian Journal of Applied Animal Science*, vol. 6, no. 3, pp. 571–579, 2016.

[9] J. Li, J.-Y. Lee, and L. Liao, "Detecting de novo plasmodesmata targeting signals and identifying pd targeting proteins," in *International Conference on Computational Advances in Bio and Medical Sciences*. Springer, 2019, pp. 1–12.

[10] J.-Y. Lee, X. Wang, W. Cui, R. Sager, S. Modla, K. Czymmek, B. Zybaliov, K. van Wijk, C. Zhang, H. Lu *et al.*, "A plasmodesmata-localized protein mediates crosstalk between cell-to-cell communication and innate immunity in arabidopsis," *The Plant Cell*, vol. 23, no. 9, pp. 3353–3373, 2011.

[11] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in neural information processing systems*, 1999, pp. 487–493.

[12] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[13] H. Mamitsuka, "A learning method of hidden markov models for sequence discrimination," *Journal of Computational Biology*, vol. 3, no. 3, pp. 361–373, 1996.

[14] P. Baldi and Y. Chauvin, "Smooth on-line learning algorithms for hidden markov models," *Neural Computation*, vol. 6, no. 2, pp. 307–318, 1994.

[15] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.