An Improvement in Baum Welch Algorithm - A Case Study of HSI future

Abstract

The Baum Welch Algorithm is widely used for training the parameters of Hidden Markov Models using EM technics. However, the traditional BW algorithm doesn't make any distributional assumption for the observations. In this paper, we propose a Distribution Based BW algorithm and study its advantages by experiments of simulated HMM chains. We also apply this new BW algorithm on HSI futures study. The new proposed Distribution Based BW algorithm can improve the performance significantly both in experiments and real-world practice.

1 Introduction

Hidden Markov Models (HMM)[1] serve as the extension of Markov chain which implies that states before the current state have no impact on the future except via the current state. HMM is composed of two parts: hidden states and observed events. An state can be transferred to another state and an observation is implied by a certain state.

An HMM is specified by the following definitions:

$$Q = q_1 q_2 \cdots q_N \quad \text{a set of N states} \tag{1}$$
$$O = o_1 o_2 \cdots o_T \quad \text{a set of T observations} \tag{2}$$
$$P = p(h, h^*) \quad \text{a transition prob matrix, as the prob of state } h \text{ to state } h^* \tag{3}$$
$$E = e(h, o_t) \quad \text{an emission prob matrix, as the prob of state } h \text{ to the observation } t \tag{4}$$
$$\pi = \pi(1)\pi(2)\cdots\pi(N) \quad \text{the initial prob matrix for states} \tag{5}$$

HMM is widely used in the real world.For example , the decoding problem is meant for decoding the observations to the hidden states. The exact application of it can be the segmentation, by selecting the most probable track of hidden states. This can be achieved by a dynamic programming method named Viterbi Algorithm[1].

In other ways, the learning algorithm and evaluation algorithm can be applied in the recognition of audio signals with observations as audio clips and hidden states as words. Also it was initially used as a landmark method in machine translation problems with source language as observations and target language as hidden states. The learning algorithm can be achieved by Baum-Welch algorithm, which is a naive form of EM algorithm. Evaluation is completed by forward and backward algorithm.

2 The Original Baum-Welch Algorithm

When learning the parameters of an HMM, we mean given an observation sequence O and the set of possible states in the HMM, learn the transfer and emission matrices. Served as a variation of EM, B-W algorithm is an iterative algorithm, computing an initial estimate for the probabilities, then using those estimates to computing a better estimate, and so on. The iteration can be illustrated as follow.

Firstly, the iterative method for α and β can be achieved by $forward$ and *backward* algorithms[1].
$forward$ variables:

$$\alpha(0, h) = \pi(h)e(h, O_0) \tag{6}$$

$$\begin{aligned}\alpha(i+1, h) &= P[O_{\leq i+1} = o_{\leq i+1}, H_{i+1} = h] \\ &= \sum_{h^* \in H} \alpha(i, h^*)p(h^*, h)e(h, o_{i+1})\end{aligned} \tag{7}$$

backward variables:

$$\beta(n, h) = P[O_{>n} = o_{>n}, H_n = h] = 1 \tag{8}$$

$$\begin{aligned}\beta(i-1,h) &= P[O_{>i-1} = o_{>i-1}|H_{i-1} = h] \\ &= \sum_{h^* \in H} p(h,h^*)e(h^*,o_i)\beta(i,h^*)\end{aligned} \tag{9}$$

Let $N(h)$ denote the number of times $H_0 = h$, $N(h,h^*)$ the number of times trasferring from h to h^*, $N(h,o)$ the number of times trasferring from state h to observation o.

Then we can use α, β, p, e to represent the above components:

$$E\{N(h)\} = P[H_0 = h|O = o, \theta]) = \frac{\alpha(0,h)\beta(0,h)}{P[O=0|\theta]} \tag{10}$$

$$\begin{aligned}E\{N(h,h^*)\} &= \sum_{i=1}^{n-1} P(H_{i+1} = h^*, H_i = h|O = o, \theta) \\ &= \frac{\sum_{i=1}^{n-1} \alpha(i,h)p(h,h^*)e(h^*,o_{i+1})\beta(i+1,h^*)}{P(O=o|\theta)}\end{aligned} \tag{11}$$

$$\begin{aligned}E\{N(h,o)\} &= \sum_{i=1}^{n-1} P(H_i = h|O = o, \theta) \\ &= \sum_{i:O_i=o} \frac{\alpha(i,h)\beta(i,h)}{P[O=o|\theta]}\end{aligned} \tag{12}$$

The log-likelihood function for $\pi(h), p(h,h^*), e(h^*,o_i)$ can ce expressed as follow:

$$l = \sum_{h\in H} N(h)log[\pi(h)] + \sum_{h\in H}\sum_{o\in\varepsilon} E(N(h,o))log[e(h,o)] + \sum_{h\in H}\sum_{h^*\in H} E(N(h,h^*))log[p(h,h^*)] \tag{13}$$

We set

$$\begin{aligned}&Q(\theta|\theta^{(t)}) = E[l(\theta|H)|O,\theta^{(t)}] \\ &= \sum_{h\in H} E(N(h)|\theta^{(t)})log[\pi(h)] + \sum_{h\in H}\sum_{0\in\varepsilon} E(N(h,o)|\theta^{(t)})log[e(h,o)] \\ &\quad + \sum_{h\in H}\sum_{h^*\in H} E(N(h,h^*)|\theta^{(t)})log[p(h,h^*)]\end{aligned} \tag{14}$$

Hence we can use EM algorithm, by Lagrange methods to maximize $Q(\theta|\theta^{(t)})$ to compute the parameters iteratively. The iterate euqations are written as follow.

$$\pi^{(t+1)}(h) = \frac{E[N(h)|\theta^{(t)}]}{\sum_{h^*\in H} E[N(h^*)|\theta^{(t)}]} \tag{15}$$

$$p(h,h^*)^{(t+1)} = \frac{E[N(h,h^*)|\theta^{(t)}]}{\sum_{h^{**}\in H} E[N(h,h^{**})|\theta^{(t)}]} \tag{16}$$

$$e(h,o)^{(t+1)} = \frac{E[N(h,o)|\theta^{(t)}]}{\sum_{o\in\varepsilon} E[N(h,o)|\theta^{(t)}]} \tag{17}$$

3 Distribution Based Baum-Welch Algorithm

When learning the parameters of emission algorithm, we simply iterate the computation and get iterated numeric results. However, sometimes we precisely know that the emission probability is conformed to a certain distribution. This is often the case. When tossing the dice, a fair dice can only result in 6 numbers and their probs are equally $\frac{1}{6}$; when changing to an unfair dice with three surfaces as 6 and the rest 1, 2, 3, the probs are $\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{2}\}$. For the former situation, we naturally want the learned emission probs to be close to uniform distribution. In continuous situations, suggest there is a cafeteria open in both sunny days and rainy days. We know the number of customers in a certain time period is conformed to Poisson distribution; in sunny days the expectation should be higher than it in rainy days. So that we can model a Hidden Markov Model with $P(customers|weather)$, the emission probs conformed to Poisson distribution, which is more reasonable and comprehensible, and improves the goodness of fit.

The iterated process can be modified as follow. After updating the $e(h,o)^{(t+1)}$, we impose the specified F distribution on it with the plug-in parametric inference. Suppose the real emission probs $e(h,o) \sim F(\theta)$, (17) is going to be rewritten as

$$\begin{aligned} e(h,o)^{(t+1)} &= \frac{E\{N(h,o)|\theta^{(t)}\}}{\sum_{o^*} E\{N(h,o^*)|\theta^{(t)}\}} \\ \hat{\theta}_h &= \theta(O) \\ e(h,o)^{(t+1)} &= f(o;\hat{\theta}_h) \end{aligned} \tag{18}$$

Suggest the distribution is $Poisson(\lambda)$, then we can redemonstrate the appended part as a more clear form,

$$\begin{aligned} \bar{O}_h^{(t+1)} &= \sum_{k=1}^{K} e(h,o_k)^{(t+1)} o_k \\ e(h,o)^{(t+1)} &= \frac{\bar{O}_h^{(t+1)^o}}{o!} e^{-o} \end{aligned} \tag{19}$$

4 Experiments of Simulated Poisson HMM Chains with 3 Hidden States

4.1 Initialization of Simulation

In this section, we are going to do some experiments to testify that the Distribution Based BW algorithm (new BW or parametric BW for short) can perform better than the non-Distribution Based BW algorithm (traditional BW for short). We also find out that the Distribution Based BW algorithm can achieve results similar to parameters estimation in Mixture Poisson Models using EM algorithm.

Firstly, we generate some 3 Hidden States - Poisson HMM chains. For convenience, the transition matrix is fixed and described in figure 1 (left). We only modify the parameters of 3 different Poisson Emissions. For instance, if we set $\{\lambda_1 = 6, \lambda_2 = 9, \lambda_3 = 15\}$, the emission values and their probabilities of each hidden states will be shown in figure 2. We can also take a look at the HMM chain shown in Figure 4.1(right) and Figure 4.3. They are irregular and unpredictable at the first glance.

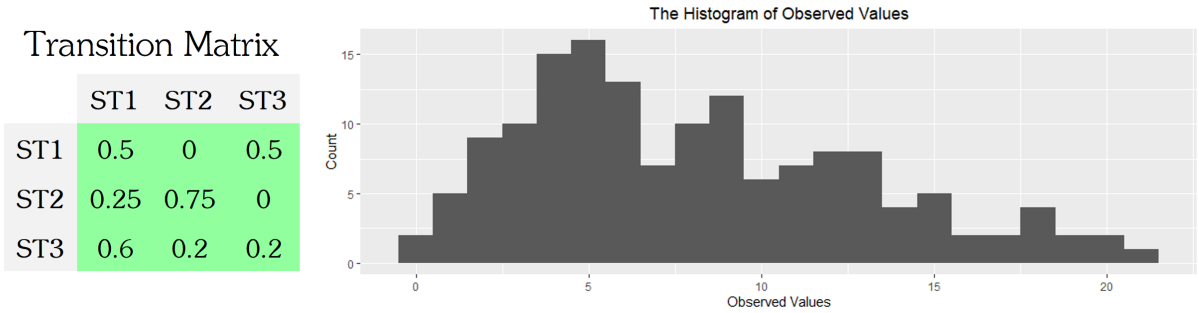

Figure 4.1: Transition Matrix and Histogram of Observations

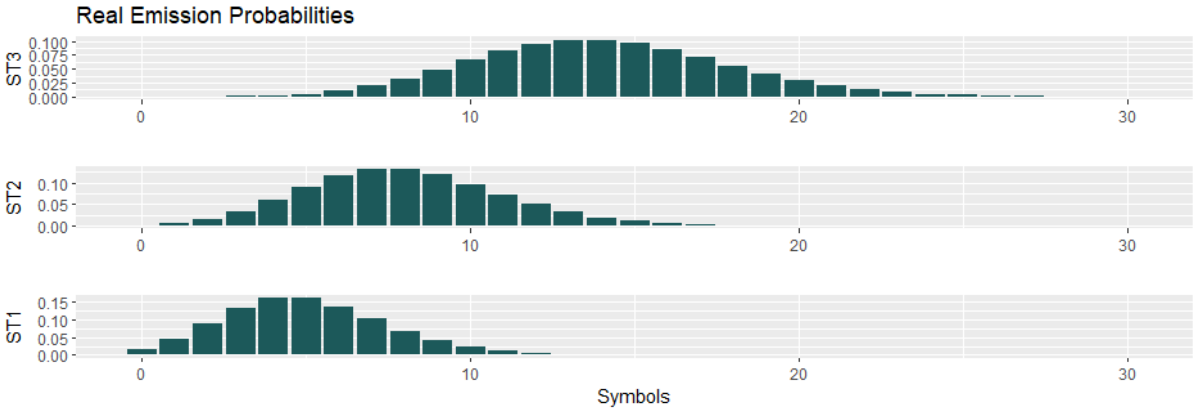

Figure 4.2: Real Emission Probabilities

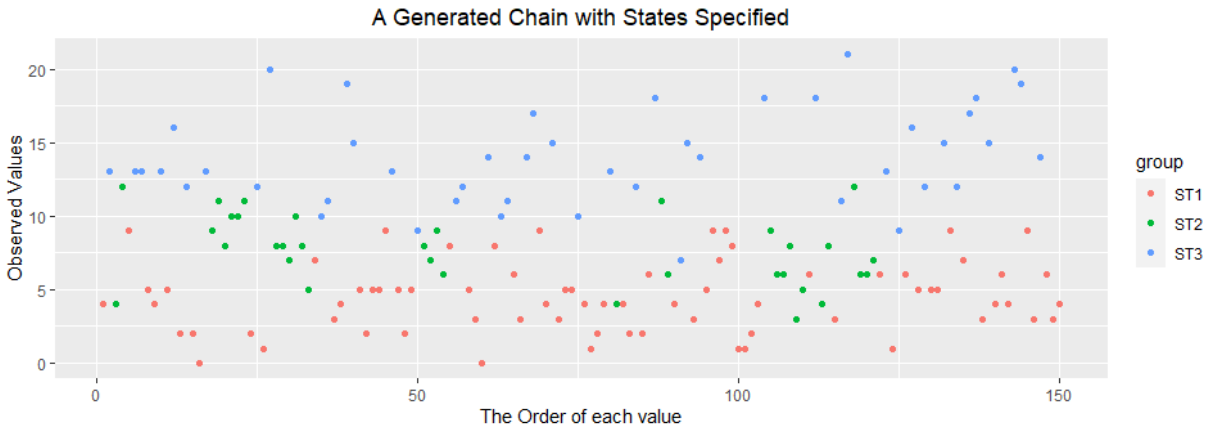

Figure 4.3: Observations of Different States

4.2 Better Transition Matrix Prediction

The advantage of the Distribution Based BW algorithm is that it can make use of the distributional information to make more accurate predictions, if we know the (emission) distribution family of each hidden state in advance. We did a lot of experiments with different combination of $\{\lambda_i | i = 1, 2, 3\}$ and find out that the new BW algorithm can outperform the traditional BW algorithm in most cases, regarding estimating the transition matrix.

	μ1 μ2 μ3 1 6 15			μ1 μ2 μ3 1 9 15			μ1 μ2 μ3 3 6 15			μ1 μ2 μ3 3 9 15			μ1 μ2 μ3 3 12 15			μ1 μ2 μ3 6 9 15			μ1 μ2 μ3 6 12 15			μ1 μ2 μ3 9 12 15		
Traditional	0.18	0	0.82	0.15	0	0.85	0.1	0	0.9	0.36	0	0.64	0.07	0	0.93	0.26	0	0.74	0	0	1	0.4	0	0.6
BW	0.12	0.88	0	0.08	0.92	0	0.26	0.74	0	0.1	0.9	0	0.18	0.82	0	0.06	0.91	0.03	0.24	0.76	0	0.15	0.85	0
prediction	0.91	0.09	0	0.77	0.1	0.13	0.69	0.13	0.18	0.85	0.15	0	0.53	0.15	0.33	0.8	0.2	0	0.52	0.15	0.33	0.44	0.56	0
Real	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5	0.5	0	0.5
Transition	0.25	0.75	0	0.25	0.75	0	0.25	0.75	0	0.25	0.75	0	0.25	0.75	0	0.25	0.75	0	0.25	0.75	0	0.25	0.75	0
Matrix	0.6	0.2	0.2	0.6	0.2	0.2	0.6	0.2	0.2	0.6	0.2	0.2	0.6	0.2	0.2	0.6	0.2	0.2	0.6	0.2	0.2	0.6	0.2	0.2
New	0.49	0	0.51	0.49	0	0.51	0.46	0	0.54	0.49	0	0.51	0.49	0.01	0.5	0.46	0	0.54	0.45	0	0.55	0.11	0.12	0.77
BW	0.25	0.75	0	0.29	0.71	0	0.25	0.73	0.01	0.26	0.74	0	0	0.67	0.33	0.26	0.74	0	0.22	0.73	0.05	0.55	0.45	0
Prediction	0.64	0.15	0.21	0.58	0.2	0.21	0.6	0.18	0.21	0.63	0.17	0.21	0.59	0.12	0.29	0.47	0.34	0.19	0.46	0.3	0.24	0.2	0.44	0.36

Figure 4.4: Distribution Based BW versus Traditional BW Predicted Transition Matrix

Figure 4.4 is a direct presentation of the data from some of our experiments. If we fixate the real transition matrix (green matrix) and only change the real emission matrix (or you can say the combinations of) to generate different HMM observation chains, both the traditional and the new BW algorithm will derive different predictions based on each HMM chain. According to figure 4, the predicted transition matrixes from the new BW algorithm are closer to the ground truth.

Figure 4.5 illustrates the comparisons between the two BW algorithms in a more "scientific" way. It is not hard to imagine that if the emission distributions of the three hidden states are very close to each other (i.e. all the hidden states emit values with nearly the same probabilities), it would be extremely difficult to distinguish these hidden states, and hence results in a bad transition matrix prediction. So, we need to measure the diversity of these emission distributions, in another word, the dispersion of $\{\lambda_i | i = 1, 2, 3\}$. Herein we simply use the standard deviations of $\{\lambda_i | i = 1, 2, 3\}$ to quantify the dispersion, and compare the performance of the new BW and the traditional BW under different situations regarding the dispersion.

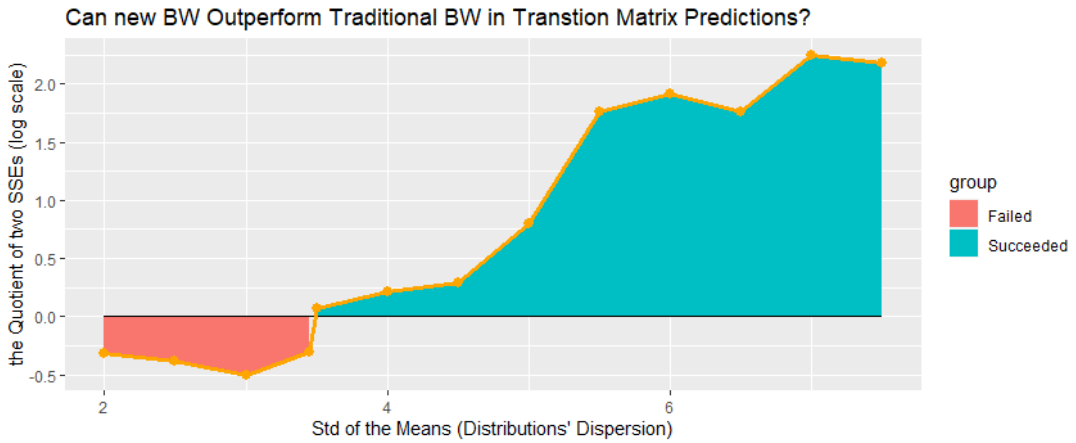

Figure 4.5: Distribution Based BW performs better with high dispersion

The dependent variable in Figure 4.5 is $\{log\frac{SSE(traditionalBW)}{parametricBW}\}$. The term $\{SSE(traditionalBW)\}$ simply means the sum of squared errors between items in the real transition matrix and that in the transition matrix predicted by traditional BW. If $\{SSE(traditionalBW)\}$ is smaller than $\{SSE(parametricBW)\}$, the term $\{SSE(traditionalBW)\}$ will be smaller than 0, indicating than traditional BW perform better than parametric BW, and vice versa.
According to Figure 4.5, we can draw a conclusion that if the diversity of emission distributions is too small, the new BW algorithm perform worse than the traditional BW algorithm in Transition Matrix Prediction. But if the diversity is above a certain threshold, which is more common, the new BW algorithm can outperform the traditional BW.

Therefore, the Distribution Based BW algorithm can outperform the traditional BW algorithm in most cases, regarding estimating the transition matrix.

4.3 Better Emission Matrix Prediction

Although the traditional BW algorithm can predict the transition matrix good enough to some extent according to Figure 4.4, it does terribly in emission matrix prediction. Figure 4.6 is a general case showing that the emission distributions predicted by the traditional BW are completely irregular, indicating that the traditional BW cannot capture any distributional pattern of emissions. While the new BW perform very well in Emission Matrix Prediction with additional distributional information.

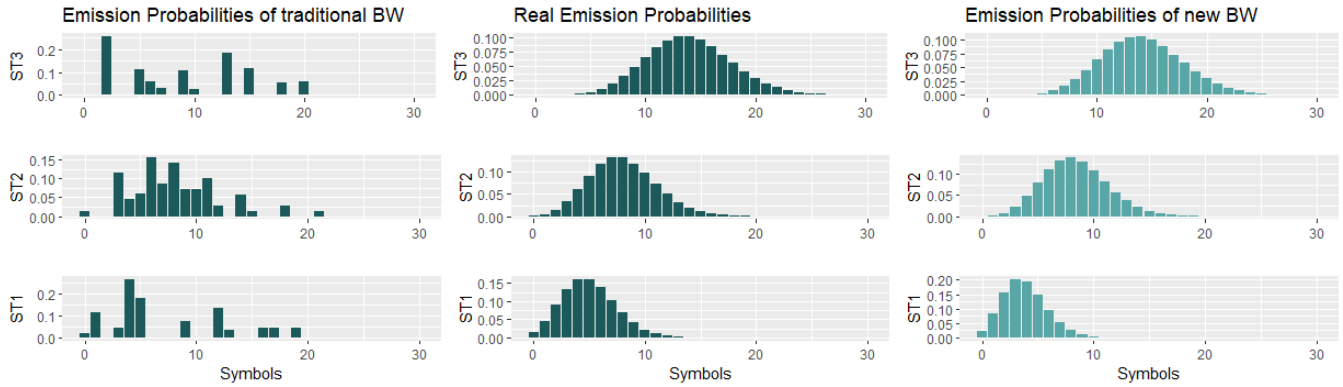

Figure 4.6: Predicted Emission Probabilities by 2 BW methods versus True Emission Probabilities

4.4 Faster Convergence and Overfitting Avoidance

In each iteration, BW algorithm allow us to predict and estimate a new set of transition matrix and emission matrix. And based on these predicted probabilities, we can calculate the probability of regenerating the observed HMM chain, based on the forward or the backward algorithm. Figure 4.7 showcases how these probabilities change through iterations. (It is one particular case, but basically all the experiment cases share the same pattern).

We surprisingly find out that the new BW algorithm performs much worse than the traditional BW algorithm in the sense of the observational probability (the magnitude of 10e-185 comparing to the magnitude of 10e-171), given that it has done so well in hidden matrix predictions. One explanation is that the traditional BW $overfits$ the observed HMM chain.

We can also view Figure 4.7 as a convergence curve. According to our experiments, the new BW never converge slower than the traditional BW.

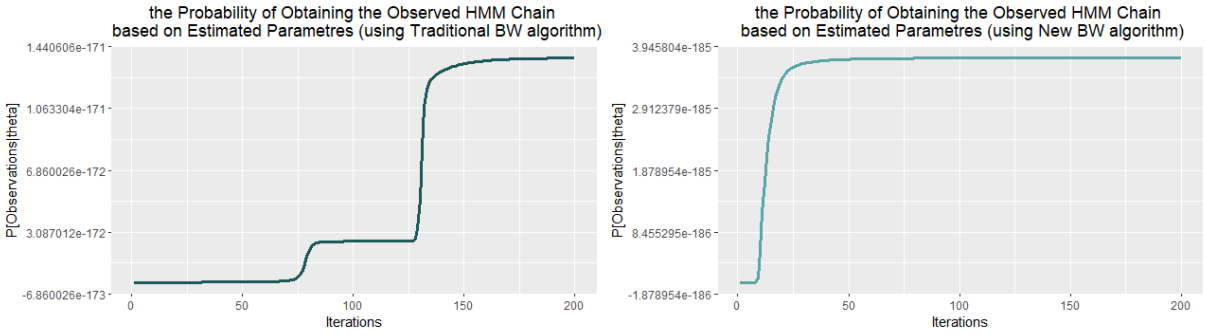

Figure 4.7: Distribution Based BW versus Traditional BW Convergence Curves

4.5 Prediction Resemblance with EM Mixture Models

Now we focus on the histogram of the generated HMM chain, Figure 4.8, which is the same figure as Figure 4.1 (right). Without considering the underlying states, we can also use the simple Poisson Mixture Model to fit the histogram. The method for such a parameters estimation is an EM algorithm, returning 3 different Poisson distributions with different $\{\lambda_i | i = 1, 2, 3\}$.

The new BW can also return such a set $\{\lambda_i | i = 1, 2, 3\}$ and constitute the emission matrix. Figure 4.9 shows that their estimations of $\{\lambda_i | i = 1, 2, 3\}$ are somehow close to each other. Such a finding is not surprising because we can actually regard the mixture model as a special case of HMM model (considering the items in the transition matrix is fixed and share the same values for every rows.)

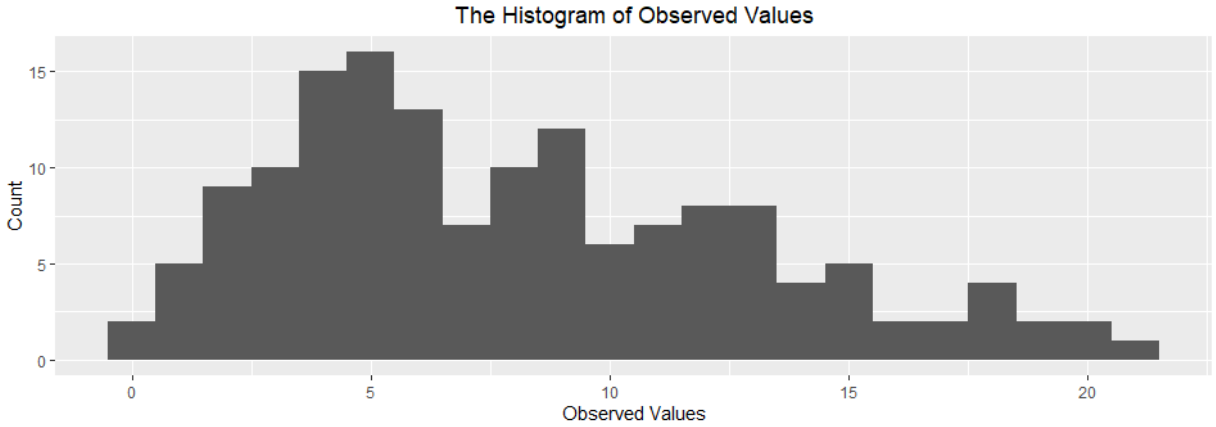

Figure 4.8: Histogram of Observations

	μ1 μ2 μ3	μ1 μ2 μ3	μ1 μ2 μ3	μ1 μ2 μ3	μ1 μ2 μ3	μ1 μ2 μ3	μ1 μ2 μ3
	1 6 15	1 9 15	3 6 15	3 9 15	3 12 15	6 9 15	6 12 15	9 12 15
new BW	0.68 4.57 13.7	0.63 7.63 13.9	2.22 4.49 13.7	2.42 8.06 13.8	2.27 9.39 13.6	3.85 8.37 14.1	3.77 9.99 13.9	7.2 10.5 13.1
EM Mixture	0.83 8.94 16.5	0.64 9.36 15.8	2.52 9.28 15.3	2.29 9.19 15.6	2.03 9.61 14.9	3.9 9.08 16.6	3.9 9.08 16.6	4.96 10.1 14.2

Figure 4.9: Distribution Based BW versus EM Mixture Models Predicted Transition Matrix Emission Probabilities

Since HMM is a generalized version of Mixture Models, it motivates us to apply the new BW algorithm in real world cases. We can recognize some mixture model's patterns and "plug-in" the distributional assumption of Emission Matrix in parametric BW, expect that the Distribution Based HMM can outperform both the non-Distribution Based HMM and the simple Mixture Model.

5 Distribution Based Baum-Welch in Real Case - A Study of HSI Futures

5.1 Data description and prepossessing

5.1.1 Data description

In order to analyse the real case study, we use HSI future as our dataset which contains the *open*, *high*, *low*, *close*, *volume*, turnover data for many indexes such as hsi, hscei, etc. In algo-trading, there is a well-known strategy: the market opens, we short HSI futures, and then close the position before the market closes. The idea is that in general the market tends to be overly excited in the morning and this enthusiasm tends to fade away in the afternoon. We apply this strategy to the HSI futures and obtain daily profit/loss as follows.

$$DailyProfit = (Open–Close–slippage) \times multiplier$$

We specific $slippage = 1.5$ and $multiplier = 50$ in this case. In this project, we only use daily profit as our training/testing object. Using an Hidden Markov model to depict this data is appropriate for the following reasons.

- The hidden state of time point=t is related to the hidden state of time point=t-1. The rise and fall of stocks are often influenced by historical trends.
- From Figure 5.2 and 5.3, we can see that Daily Profit's state transitions are frequent on the time line, and the Observation distribution may conform with a mixture model.
- To fully present the mix states of Daily Profit, we use R package *mixtools* to fit a 2 component gaussian mixture model as Figure 5.1 shows, and the fitted density curves show the same traits we have mentioned above.

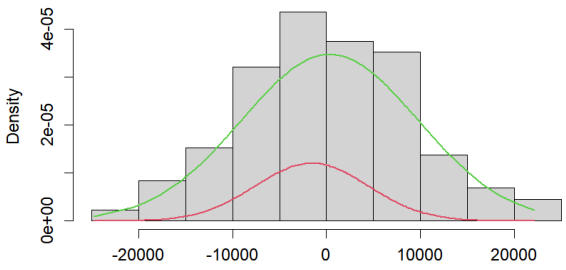

Figure 5.1: 2-Component Gaussian Mixtures

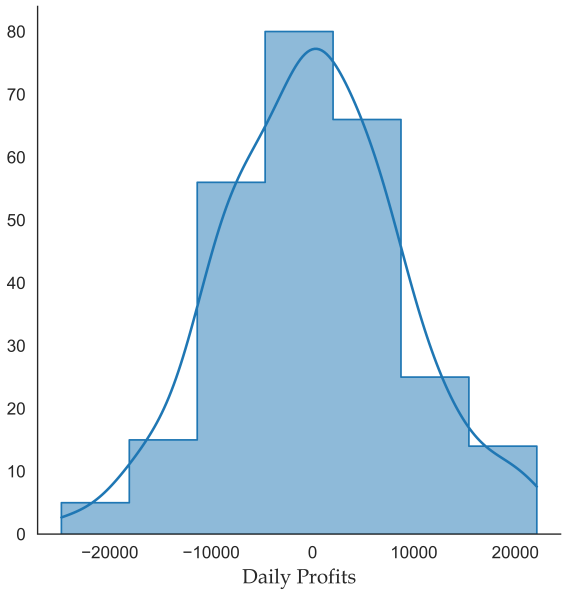

Figure 5.2: Emission Distribution parametric

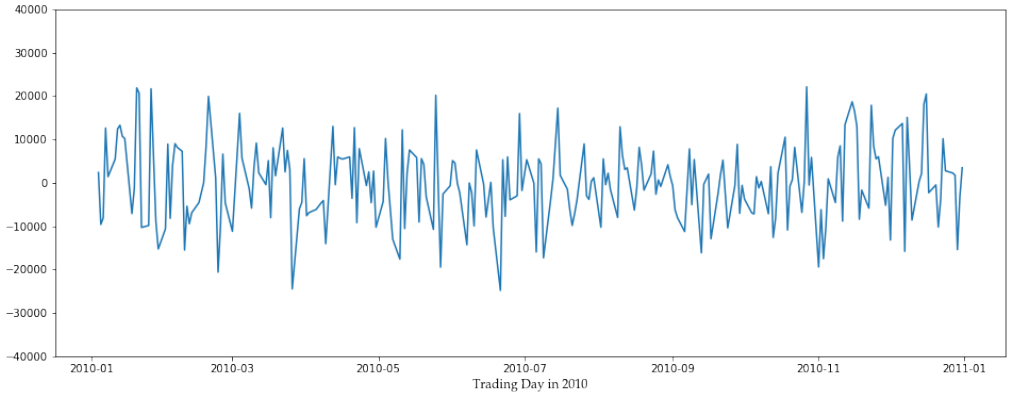

Figure 5.3: 2-Component Gaussian Mixtures

5.1.2 Prepossessing

Our observation is a time series which leads to an extraordinarily huge emission matrix. In addition, the high dimension means data becomes quite sparse in the space so the training result would demonstrates invalid 'NAs' since the denominator is easily becomes zero during training. To solve this issue, we force the data to be discretized by dividing finite bins and use mid-value of each class as our observation. After large test on our dataset, we choose seven bins and two distinct states for win and loss.

5.2 Methodology

After initializing with the same random number seed, we use the traditional Baum Welch method and the Distribution Based Baum Welch method to estimate the Initial Probability, Transition Matrix, and Emission Matrix, respectively. As mentioned in Chapter 3, the premise of the distribution we added to the Emission Matrix estimation of the Distribution Based Baum Welch method, in this experiment, we subjectively suppose that Daily Profit follows a Gaussian mixture model in two states. After updating the $e(h,o)^{t+1}$, we impose a Gaussian distribution on it with the plug-in parametric inference as follows.

$$E[O_h^{t+1}] = \sum_{k=1}^{K} e(h,o_k)^{(t+1)} \cdot o_k = \mu_h^{(t+1)}$$

$$Var[O_h^{t+1}] = \sum_{k=1}^{K} e(h,o_k)^{(t+1)} \cdot (o_k - E[O_h^{t+1}]) = \sigma_h^{2(t+1)}$$

$$e(h,o_k)^{(t+1)} = \frac{1}{\sqrt{2\pi}\sigma_h^{(t+1)}} \cdot exp(-\frac{(o_k - \mu_h^{(t+1)})^2}{2\sigma_h^{2(t+1)}})$$

Noted that in this experiment, we used the the standard deviation clip method in order to prevent the emission probability of a State from being 1 to a single Observation, and the other being 0, resulting in a sigma of 0, which makes the Emission Matrix unable to be updated. That is:

$$Var[O_h^{t+1}] = max(Var[O_h^{t+1}], 1)$$

5.3 Real case study

5.3.1 Training and Model's Parameters

For real case study, we adopts 2010 HSI daily profit as our dataset which mentioned in 5.1.1. After training them in traditional and Distribution Based Baum Welch algorithms repectively, we get two groups of model.Their results are shown as follow:

Traditional Baum Welch
Initial probability: $\pi = (0, 1)$

Transition matrix P:
$p(State1|State1) = 0$, $p(State2|State1) = 1$
$p(State1|State2) = 0$, $p(State2|State2) = 1$

Emission matrix E:
$p(-22060|ST_1) = 0.1550$, $p(-22060|ST_2) = 0.1330$
$p(-14123|ST_1) = 0.0462$, $p(-14123|ST_2) = 0.1010$
$p(-6187|ST_1) = 0.2698$, $p(-6187|ST_2) = 0.1759$
$p(1750|ST_1) = 0.0211$, $p(1750|ST_2) = 0.2558$
$p(9687|ST_1) = 0.1609$, $p(9687|St_2) = 0.1442$
$p(17623|ST_1) = 0.1248$, $p(17623|ST_2) = 0.1525$
$p(25560|ST_1) = 0.2221$, $p(25560|ST_2) = 0.0376$

Parametric Baum Welch
Initial probability: $\pi = (1, 0)$

Transition matrix P:
$p(State1|State1) = 0.8330$, $p(State2|State1) = 0.1169$
$p(State1|State2) = 0.5954$, $p(State2|State2) = 0.4046$

Emission matrix E:
$p(-22060|ST_1) = 0.0178$, $p(-22060|ST_2) = 0.000$
$p(-14123|ST_1) = 0.1270$, $p(-14123|ST_2) = 0.0033$
$p(-6187|ST_1) = 0.3435$, $p(-6187|ST_2) = 0.0485$
$p(1750|ST_1) = 0.3527$, $p(1750|ST_2) = 0.2411$
$p(9687|ST_1) = 0.1374$, $p(9687|ST_2) = 0.4135$
$p(17623|ST_1) = 0.0203$, $p(17623|ST_2) = 0.2440$
$p(25560|ST_1) = 0.0011$, $p(25560|ST_2) = 0.0496$

Noted that π means an initial probability matrix over all states. The i^{th} entry in the matrix is the probability that the Markov chain will start in State i. Some States j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, it should satisfy $\sum_{i=1}^{n} \pi_i = 1$. P means a transition probability matrix,each $a_{i,j}$ representing the probability of moving from State i to state j, s.t.$\sum_{j=1}^{N} a_{i,j} = 1$. E is a sequence of observation likelihoods, also called emission probabilities, each expressing the probability of an observation O_t being generated from a hidden State i.

5.3.2 Evaluation

Evaluation problem can be used for isolated recognition.– Given HMM parameters & observation sequence $\{O_t\}_{t=1}^{T}$ find probability of observed sequence $p(\{O_t\}_{t=1}^{T})$.

In order to test the feasibility and accuracy of our Distribution Based Baum Welch (abbreviated as BW hereinafter), we use the 2010 HSI daily profit/loss as the training dataset to generate two models based on traditional and Distribution Based BW respectively and calculate the observed sequence probability based on 2011 HSI daily profit/loss. The larger the possibility of observed sequence, the better the model we achieve.

5.3.3 Similarity

From the 4.3.4 results, it demonstrates that for different BWs, their emission matrices are inclined to the two symbols which in the middle of matrix (-6187 and 1750), which also shows the feasibility of our distribution based algorithm since in reality, the profit and loss of the daily income is generally not too large, It is common for profit and loss to be concentrated around zero.

5.3.4 Comparison

From the training result in figures below, it is obvious the emission matrix of Distribution Based BW algorithm consists of two states, while the traditional one is in chaos which is impossible for us to illustrate the distribution of hidden variables. Meanwhile, from the parametric emission matrix, we believe that these two hidden states represent a bull market and a bear market respectively, because their distributions are located on positive and negative axes respectively. Also, from the results in 5.3.1, we could find that for traditional BW algorithm, the transition matirx is meaningless as the hidden states always be state 2 whereas in Distribution Based BW, it is possible to enter different states with different probabilities.

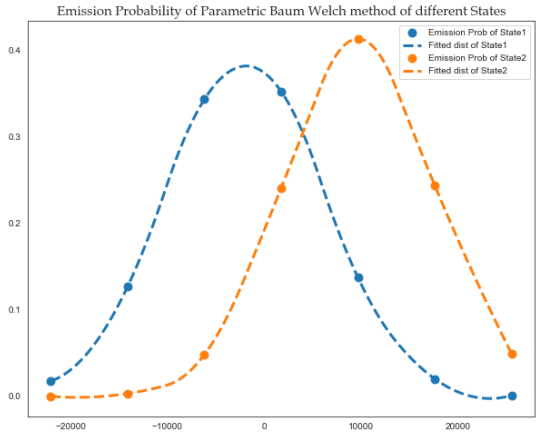

Figure 5.4: New BW Emission Distribution

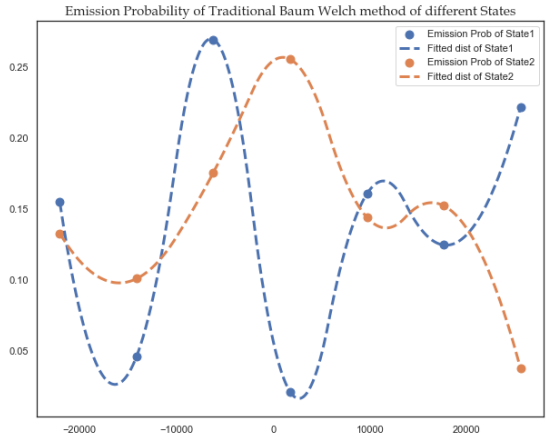

Figure 5.5: Traditional Emission Distribution

The next step we use the 2011 HSI daily profit/loss to estimate the probability of the observation sequence based on these two models. The evaluation result shows that the Distribution Based BW is better than traditional one as the former's probability ($1e-14$) is far greater than the latter's ($1e-16$). However, we can observe that the traditional BW convergence rate is slightly faster than the new BW's which gives a contradiction to 4.4. We guess it related to the transition matrices as the traditional one always hit on the second state whereas the hidden states of distribution based model is unstable which leads to different convergence rate. Besides, the Distribution Based BW has an obvious falling process for the estimation of the observation probability of 2011 daily profit, which may be caused by the influence of different random initialization.

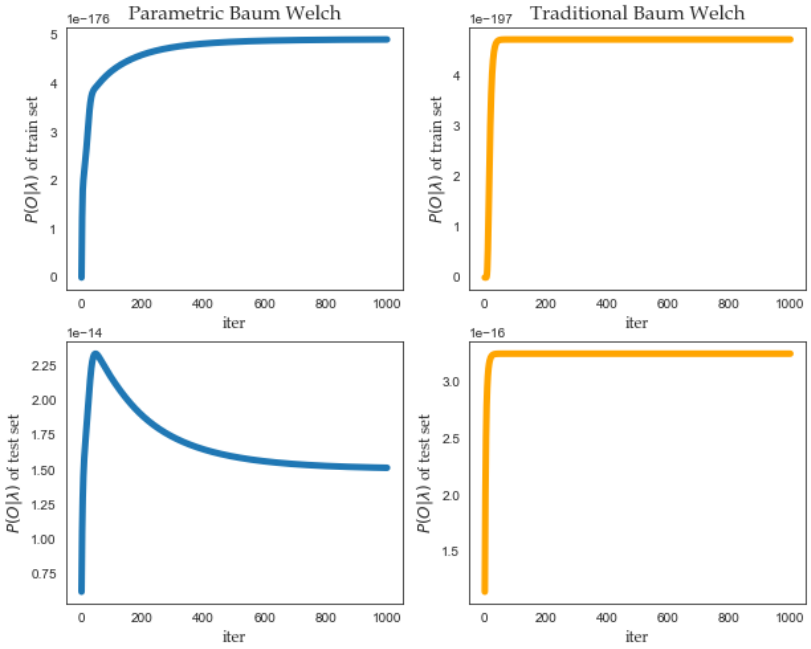

Figure 5.6: Convergence

References

[1] Givens, G. H., & Hoeting, J. A. (2012). Computational statistics (Vol. 703). John Wiley & Sons.

[2] Tang, H., Hasegawa-Johnson, M., & Huang, T. S. (2010, March). Toward robust learning of the Gaussian mixture state emission densities for hidden Markov models. In 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 5242-5245). IEEE.

[3] Bilmes, J. A. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. International Computer Science Institute, 4(510), 126.

6 Appendix

6.1 R Codes

We mainly use R Program to achieve the Parametric and Traditional Baum Welch method

```
#############################################################################################
###################################   Define Functions   ####################################
#############################################################################################

# initialize
ranProb = function(n, method='uniform') {
  if (method=='uniform'){
    p = runif(n)
  }
  else if(method=='Gaussian'){
    p = rnorm(n)
  }
  # add method you like...
  return (p / sum(p))
}


# len_state means the number of unique States
get_initProb = function(States, method = 'uniform'){
  len_state = length(States)
  res = matrix(ranProb(len_state, method = 'uniform'), nrow = 1, byrow = TRUE, dimnames = list(1,
    States))
}

get_initTran = function(States, method = 'uniform'){

  len_state = length(States)
  initTran_vec = c()

  for (i in 1:len_state){
    initTran_vec = append(initTran_vec, ranProb(len_state, method=method))
  }
  res = matrix(initTran_vec, nrow=len_state, byrow=TRUE, dimnames = list(States,States))
  return(res)
}


get_initEmiss = function(States, Observation, method = 'uniform'){

  len_state = length(States)
  len_obs = length(unique(Observation))
  initEmiss_vec = c()
  for (i in 1:len_state){
    initEmiss_vec = append(initEmiss_vec, ranProb(len_obs, method = method))
  }
  res = matrix(initEmiss_vec, nrow=len_state, byrow=TRUE, dimnames = list(States,sort(unique(
    Observation))))
  return(res)
}



# compute alpha & beta
compute_alpha = function(Observation, States, Emission, Transition, initProb){

  # initial alpha matrix nrow=T(length of Observation), ncol=length(States)
  alpha = matrix(0, nrow=length(Observation), ncol=length(States), dimnames = list(seq(1,length(
    Observation)),States))

  # Compute alpha_t(j) when t == 1
  for(j in 1:length(States)){
    alpha[1, j] = Emission[j, toString(Observation[1])]*initProb[1, j]
  }

  # Use dp equation to compute alpha_t(j)
  for(t in 2:length(Observation)){
    for(j in 1:length(States)){
      alpha[t, j] = sum(Emission[j, toString(Observation[t])] * Transition[,j] * alpha[t-1, ])
    }
  }

  return(alpha)
}


```

```r
compute_beta = function(Observation, States, Emission, Transition, initProb){

  # initial beta matrix nrow=T(length of Observation), ncol=length(States)
  beta = matrix(0, nrow=length(Observation), ncol=length(States), dimnames = list(seq(1,length(
    Observation)),States))

  # Compute beta_t(i) when t == length(Obsevation)
  for(i in 1:length(States)){
    beta[length(Observation),i] = 1
  }

  # Use dp equation to compute beta_t(i)
  for(t in (length(Observation)-1):1){
    for(i in 1:length(States)){
      beta[t,i] = sum(Emission[ , toString(Observation[t+1])] * Transition[i, ] * beta[t+1, ])
    }
  }

  return(beta)
}

# Define a traditional Baum Welch
Baum_Welch = function(iteration, initProb, Transition, Emission, Observation,
                      States, method='Tradition', distribute = 'Gaussian',predict_data){
  ######################################## train part ########################################
  # Get evaluation data
  test_obs = predict_data$labels[1:20]

  # collect alpha and check the convergence
  alpha_train = c()
  alpha_test = c()

  # collect sample mean and sample standard deviation
  mean_list = matrix(0, nrow = length(States), ncol=iteration, dimnames = list(States, 1:iteration))
  sd_list = matrix(0, nrow = length(States), ncol=iteration, dimnames = list(States, 1:iteration))


  for(iter in 1:iteration){
    new_initProb = matrix(0, nrow=1, ncol=ncol(Transition), byrow=TRUE, dimnames = list(1,States))

    new_Transition = matrix(0, nrow=nrow(Transition), ncol=ncol(Transition), byrow=TRUE, dimnames =
    list(States,States))

    new_Emission = matrix(0, nrow=nrow(Emission), ncol=ncol(Emission), byrow=TRUE, dimnames = list(
    States,colnames(Emission)))

    alpha = compute_alpha(Observation, States, Emission, Transition, initProb)

    beta = compute_beta(Observation, States, Emission, Transition, initProb)

    P_O_lambda = sum(alpha[1, ]*beta[1, ])


    # get new initProb matrix
    for(i in 1:ncol(initProb)){
      new_initProb[1, i] = (alpha[1, i] * beta[1, i]) / P_O_lambda
    }

    # get new Transition matrix
    for(i in 1:nrow(Transition)){
      for(j in  1:ncol(Transition)){
        epsilon = 0 # initialize epsilon
        gamma_ = 0  # initialize gamma
        for(t in 1:(length(Observation)-1)){
          epsilon = epsilon + alpha[t, i]*Transition[i, j]*Emission[j, toString(Observation[t+1])]*
    beta[t+1, j]
          gamma_ = gamma_ + alpha[t, i]*beta[t, i]
        }
        new_Transition[i, j] = epsilon / gamma_
      }
    }

    # get new Emission matrix
    for(j in 1:nrow(Emission)){
      for(k in 1:ncol(Emission)){
        numerator = 0
        denominator = 0
        for(t in 1:(length(Observation)-1)){
          if(toString(Observation[t]) == colnames(Emission)[k]){
            numerator = numerator + alpha[t, j]*beta[t, j]
          }
```

```
          denominator = denominator + alpha[t, j]*beta[t, j]
        }
        new_Emission[j, k] = numerator/denominator
      }
    }

    # update initProb, Transition, Emission
    initProb = new_initProb
    Transition = new_Transition

    # Traditional method
    if(method=='Tradition'){
      Emission = new_Emission
    }

    # Advanced Baum Welch
    if(method == 'Advanced'){
      for(i in 1:length(States)){
        mean_S = sum(new_Emission[i, ] * sort(unique(Observation)))
        sd_S = sqrt(sum((sort(unique(Observation)) - mean_S)^2 * new_Emission[i, ]))

        mean_list[i, iter] = mean_S
        sd_list[i, iter] = sd_S

        for(j in 1:ncol(Emission)){
          Emission[i, j] = dnorm(sort(unique(Observation))[j], mean=mean_S, sd=max(sd_S, 1))
        }

        # Normalize
        s = sum(Emission[i, ])
        for(j in 1:ncol(Emission)){
          Emission[i, j] = Emission[i, j]/s
        }

      }
    }
    # collect alpha and check the convergence
    alpha_train = append(alpha_train, sum(alpha[nrow(alpha), ]))

    ########################################## Test part ##########################################
    # We will do evaluation for every 5 epoch
    tmp = compute_alpha(test_obs,States,Emission,Transition,initProb)
    alpha_test = append(alpha_test, sum(tmp[nrow(tmp), ]))


  }



  res = list('initProb'=initProb, 'Transition'=Transition, 'Emission'=Emission,
             'mean_list'=mean_list, 'sd_list'=sd_list,
             'alpha_train'=alpha_train, 'alpha_test'=alpha_test)

  # Visualization
  plot(alpha_train)
  plot(alpha_test)

  plot(mean_list[1,])
  plot(mean_list[2,])
  plot(sd_list[1,])
  plot(sd_list[2,])

  return(res)
}

#########################################################################################################
########################################  Main  #########################################################
#########################################################################################################


#####################################  Learning  ########################################

# Create States and Observation
States = c("ST1", "ST2")
data = read.csv('data/data_2010.csv')
predict_data = read.csv('data/data_2011.csv')
Observation = unlist(data$labels)

# Set random seed, iteration
iteration = 1000

```

```r
# initialize init_Prob, init_Tran, init_Emiss for advanced method
set.seed(7102)
initProb = get_initProb(States) # method can be changed
Transition = get_initTran(States)
Emission = get_initEmiss(States, Observation) # unique means get number of distinct values of
    Observation

test_Advanced = Baum_Welch(iteration, initProb, Transition, Emission,
                           Observation, States, method = 'Advanced',
                           predict_data = predict_data)

# initialize init_Prob, init_Tran, init_Emiss for tradition method
set.seed(7102)
initProb = get_initProb(States) # method can be changed
Transition = get_initTran(States)
Emission = get_initEmiss(States, Observation) # unique means get number of distinct values of
    Observation
test_Traditional = Baum_Welch(iteration, initProb, Transition, Emission,
                              Observation, States, method = 'Traditional',
                              predict_data = predict_data)
# Emission plot
# Transition
# plot observation
# P(O|lambda)
#####################################  Evaluation  ##########################################

predict_data = read.csv('data/data_2011.csv')
obs = predict_data$labels[1:20]
compute_alpha(obs,States,test_Advanced$Emission,test_Advanced$Transition,test_Advanced$initProb)
```

6.2 Codes documentation

Baum Welch

Description
In order to add our improvement ideas to the Baum Welch algorithm, we build Baum Welch from scratch with R Program, and add some more convenient functions and more detailed return information on this basis so that we can better experiment to verify our ideas.

Usage
Baum_Welch(iteration, initProb, Transition, Emission, Observation, States, method = 'Advanced', predict_data)

Arguments

iteration	The number of iterations of the Baum Welch algorithm.
initProb	Initial probability π, before using the Baum Welch function, please initialize the initial probability π randomly with function **get_initProb**.
Transition	Initial Transition Matrix, before using the Baum Welch function, please initialize the Transition Matrix randomly with function **get_initTran**.
Emission	Initial Emission Matrix, before using the Baum Welch function, please initialize the Emission Matrix randomly with function **get_initEmiss**.
Observation	Please input Observation in list format $c(Obs_1, Obs_2, Obs_3, ...)$, please note that the current version only supports numeric input.
States	Please input States in list format. $c(State_1, State_2, State_3, ...)$
method	You could choose to use 'Advanced' or 'Traditional' here. 'Advanced' means the new method we mentioned in the early pages while 'Traditional' simply means the traditional Baum Welch algorithm.
predict_data	To make it more convenient to observe the difference between 2 methods, We allow users to pass in prediction data to verify the convergence and performance of the algorithm, please note that the 'predict data' here is the real data we can observe.

Value

initProb	The initial Probability π estimated by Baum Welch Algorithm that maximize the $P(O\|\lambda)$.

Transition	The Transition matrix estimated by Baum Welch Algorithm that maximize the $P(O\|\lambda)$.
Emission	The Emission matrix estimated by Baum Welch Algorithm that maximize the $P(O\|\lambda)$.
mean_list	Expectation of the estimate of the Emission Matrix distribution for each iteration.
sd_list	Standard Variation of the estimate of the Emission Matrix distribution for each iteration.
alpha_train	The $P(O\|\lambda)$ calculated by the forward algorithm after each iteration based on the training set.
alpha_train	The $P(O\|\lambda)$ calculated by the forward algorithm after each iteration based on the test set, which is the predict data.