Baum–Welch algorithm

From Wikipedia, the free encyclopedia

In electrical engineering, computer science, statistical computing and bioinformatics, the **Baum–Welch algorithm** is used to find the unknown parameters of a hidden Markov model (HMM). It makes use of the forward-backward algorithm and is named for Leonard E. Baum and Lloyd R. Welch.

History [edit]

(HMMs) and the Baum–Welch algorithm were first described in a series of articles by Leonard E. Baum and his peers at the Institute for Defense Analysis in the late 1960s.[1] One of the first major applications of HMMs was to the field of speech processing.[2] In the 1980s, HMMs were emerging as a useful tool in the analysis of biological systems and information, and in particular genetic information.[3] They have since become an important tool in the probabilistic modeling of genomic sequences.[4]

Description [edit]

A Hidden Markov Model describes the joint probability of a collection of 'hidden' and observed discrete random variables. It relies on the assumption that the i^{th} hidden variable given the $(i-1)^{th}$ hidden variable is independent of previous hidden variables, and the current observation variables depend only on the current hidden state.

The Baum–Welch algorithm uses the well known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden Markov model given a set of observed feature vectors.

Let X_t be a discrete hidden random variable with N possible values. We assume the $P(X_t|X_{t-1})$ is independent of time t, which leads to the definition of the time independent stochastic transition matrix

$$A = \{a_{ij}\} = P(X_t = j|X_{t-1} = i).$$

The initial state distribution (i.e. when $t = 1$) is given by

$$\pi_i = P(X_1 = i).$$

The observation variables Y_t can take one of K possible values. The probability of a certain observation at time t for state j is given by

$$b_j(y_t) = P(Y_t = y_t|X_t = j).$$

Taking into account all the possible values of Y_t and X_t we obtain the K by N matrix $B = \{b_j(y_t)\}$.

An observation sequence is given by $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$.

Thus we can describe a hidden Markov chain by $\theta = (A, B, \pi)$. The Baum–Welch algorithm finds a local maximum for $\theta^* = \arg\max_\theta P(Y|\theta)$. (i.e. the HMM parameters θ that maximise the probability of the observation.)[5]

Algorithm [edit]

Set $\theta = (A, B, \pi)$ with random initial conditions. They can also be set using prior information about the parameters if it is available.

Forward procedure [edit]

Let $\alpha_i(t) = P(Y_1 = y_1, ..., Y_t = y_t, X_t = i|\theta)$, the probability of seeing the $y_1, y_2, ..., y_t$ and being in state i at time t. This is found recursively:

1. $\alpha_i(1) = \pi_i b_i(y_1)$
2. $\alpha_j(t+1) = b_j(y_{t+1}) \sum_{i=1}^{N} \alpha_i(t) a_{ij}$

Backward procedure [edit]

Let $\beta_i(t) = P(Y_{t+1} = y_{t+1}, ..., Y_T = y_T | X_t = i, \theta)$ that is the probability of the ending partial sequence $y_{t+1}, ..., y_T$ given starting state i at time t. We calculate $\beta_i(t)$ as,

1. $\beta_i(T) = 1$
2. $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(y_{t+1})$

Update [edit]

We can now calculate the temporary variables, according to Bayes' theorem:

$$\gamma_i(t) = P(X_t = i|Y, \theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

which is the probability of being in state i at time t given the observed sequence Y and the parameters θ

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j|Y, \theta) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k(t) a_{kl} \beta_l(t+1) b_l(y_{t+1})} = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}$$

which is the probability of being in state i and j at times t and $t+1$ respectively given the observed sequence Y and parameters θ.

θ can now be updated:

- $\pi_i^* = \gamma_i(1)$

which is the expected frequency spent in state i at time 1.

- $a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$

which is the expected number of transitions from state i to state j compared to the expected total number of transitions away from state i. To clarify, the number transitions away from state i does not mean transitions to a different state j, but to any state including itself. This is equivalent to the number of times state i is observed in the sequence from t=1 to t=T-1.

- $b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t = v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$

where $1_{y_t = v_k} = \begin{cases} 1, & \text{if } y_t = v_k \\ 0, & \text{otherwise} \end{cases}$ is an indicator function and $b_i^*(v_k)$ is the expected number of times the output observations have been equal to v_k while in state i over the expected total number of times in state i.

These steps are now repeated iteratively until a desired level of convergence.

Note: It is possible to over-fit a particular data set. That is $P(Y|\theta_{final}) > P(Y|\theta_{true})$. The algorithm also does **not** guarantee a global maximum.

Example [edit]

Suppose we have a chicken from which we collect eggs at noon everyday. Now whether or not the chicken has laid eggs for collection depends on some unknown factors that are hidden. We can however (for simplicity) assume that there are only two states that determine whether the chicken lays eggs. Now we don't know the state at the initial starting point, we don't know the transition probabilities between the two states and we don't know the probability that the chicken lays an egg given a particular state.[6][7] To start we first guess the transition and emission matrices.

Transition

	State 1	State 2
State 1	0.5	0.5
State 2	0.3	0.7

Emission

	No Eggs	Eggs
State 1	0.3	0.7
State 2	0.8	0.2

Initial

State 1	0.2
State 2	0.8

We then take set of observations (E = eggs, N = no eggs): NN, NN, NN, NN, NE, EE, EN, NN, NN
The next step is to estimate a new transition matrix.

Observed sequence	Probability of sequence and state is S1 then S2	Highest Probability of observing that sequence
NN	0.024	0.3584 S2,S2

NN	0.024	0.3584 S2,S2
NN	0.024	0.3584 S2,S2
NN	0.024	0.3584 S2,S2
NE	0.006	0.1344 S2,S1
EE	0.014	0.0490 S1,S1
EN	0.056	0.0896 S2,S2
NN	0.024	0.3584 S2,S2
NN	0.024	0.3584 S2,S2
Total	0.22	2.4234

Thus the new estimate for the S1 to S2 transition is now $\frac{0.22}{2.4234} = 0.0908$ (referred to as "Pseudo probabilities" in the following tables). We then calculate the S2 to S1, S2 to S2 and S1 to S1 transition probabilities and normalize so they add to 1. This gives us the updated transition matrix:

Old Transition Matrix

	State 1	State 2
State 1	0.5	0.5
State 2	0.3	0.7

New Transition Matrix (Pseudo Probabilities)

	State 1	State 2
State 1	0.0598	0.0908
State 2	0.2179	0.9705

New Transition Matrix (After Normalization)

	State 1	State 2
State 1	0.3973	0.6027
State 2	0.1833	0.8167

Next, we want to estimate a new emission matrix,

Observed Sequence	Highest probability of observing that sequence if E is assumed to come from S1	Highest Probability of observing that sequence
NE	0.1344 S2,S1	0.1344 S2,S1
EE	0.0490 S1,S1	0.0490 S1,S1
EN	0.0560 S1,S2	0.0896 S2,S2
Total	0.2394	0.2730

The new estimate for the E coming from S1 emission is now $\frac{0.2394}{0.2730} = 0.8769$

This allows us to calculate the emission matrix as described above in the algorithm, by adding up the probabilities for the respective observed sequences. We then repeat for if N came from S1 and for if N and E came from S2 and normalize.

Old Emission Matrix

	No Eggs	Eggs
State 1	0.3	0.7
State 2	0.8	0.2

New Emission Matrix (Estimates)

	No Eggs	Eggs
State 1	0.0876	0.8769
State 2	1.0000	0.7385

New Emission Matrix (After Normalization)

	No Eggs	Eggs
State 1	0.0908	0.9092
State 2	0.5752	0.4248

To estimate the initial probabilities we assume all sequences start with the hidden state S1 and calculate the highest probability and then repeat for S2. Again we then normalize to give an updated initial vector.
Finally we repeat these steps until the resulting probabilities converge satisfactorily.

Applications [edit]

Speech recognition [edit]

Hidden Markov Models were first applied to speech recognition by James K. Baker in 1975.[8] Continuous speech recognition occurs by the following steps, modeled by a HMM. Feature analysis is first undertaken on temporal and/or spectral features of the speech signal. This produces an observation vector. The feature is then compared to all sequences of the speech recognition units. These units could be phonemes, syllables, or whole-word units. A lexicon decoding system is applied to constrain the paths investigated, so only words in the system's lexicon (word dictionary) are investigated. Similar to the lexicon decoding, the system path is further constrained by the rules of grammar and syntax. Finally, semantic analysis is applied and the system outputs the recognized utterance. A limitation of many HMM applications to speech recognition is that the current state only depends on the state at the previous time-step, which is unrealistic for speech as dependencies are often several time-steps in duration.[9] The Baum–Welch algorithm also has extensive applications in solving HMMs used in the field of speech synthesis.[10]

Cryptanalysis [edit]

The Baum–Welch algorithm is often used to estimate the parameters of HMMs in deciphering hidden or noisy information and consequently is often used in cryptanalysis. In data security an observer would like to extract information from a data stream without knowing all the parameters of the transmission. This can involve reverse engineering a channel encoder.[11] HMMs and as a consequence the Baum–Welch algorithm have also been used to identify spoken phrases in encrypted VoIP calls.[12] In addition HMM cryptanalysis is an important tool for automated investigations of cache-timing data. It allows for the automatic discovery of critical algorithm state, for example key values.[13]

Applications in bioinformatics [edit]

Finding genes [edit]

Prokaryotic [edit]

The GLIMMER (Gene Locator and Interpolated Markov ModelER) software was an early gene-finding program used for the identification of coding regions in prokaryotic DNA.[14][15] GLIMMER uses Interpolated Markov Models (IMMs) to identify the coding regions and distinguish them from the noncoding DNA. The latest release (GLIMMER3) has been shown to have increased specificity and accuracy compared with its predecessors with regard to predicting translation initiation sites, demonstrating an average 99% accuracy in locating 3' locations compared to confirmed genes in prokaryotes.[16]

Eukaryotic [edit]

The GENSCAN webserver is a gene locator capable of analyzing eukaryotic sequences up to one million base-pairs (1 Mbp) long.[17] GENSCAN utilizes a general inhomogeneous, three periodic, fifth order Markov model of DNA coding regions. Additionally, this model accounts for differences in gene density and structure (such as intron lengths) that occur in different isochores. While most integrated gene-finding software (at the time of GENSCANs release) assumed input sequences contained exactly one gene, GENSCAN solves a general case where partial, complete, or multiple genes (or even no gene at all) is present.[18] GENSCAN was shown to exactly predict exon location with 90% accuracy with 80% specificity compared to an annotated database.[19]

Copy-number variation detection [edit]

Copy-number variations (CNVs) are an abundant form of genome structure variation in humans. A discrete-valued bivariate HMM (dbHMM) was used assigning chromosomal regions to seven distinct states: unaffected regions, deletions, duplications and four transition states. Solving this model using Baum-Welch demonstrated the ability to predict the location of CNV breakpoint to approximately 300 bp from micro-array experiments.[20] This magnitude of resolution enables more precise correlations between different CNVs and across populations than previously possible, allowing the study of CNV population frequencies. It also demonstrated a direct inheritance pattern for a particular CNV.

Implementations [edit]

- jhmm or jahmm implementation in Java.
- HMMFit function in the RHmm package for R.
- ghmm C library with Python bindings that supports both discrete and continuous emissions.
- hmmtrain in MATLAB
- Accord.NET in C#

See also [edit]

- Viterbi algorithm
- Hidden Markov model
- EM algorithm
- Maximum Likelihood
- Speech Recognition
- Bioinformatics
- Cryptanalysis

References [edit]

1. ^ Rabiner, Lawrence. "First Hand: The Hidden Markov Model". IEEE Global History Network. Retrieved 2 October 2013.
2. ^ Jelinek, F; Bahl, L.; Mercer, R. (May 1975). "Design of a linguistic statistical decoder for the recognition of continuous speech". *IEEE Transactions of Information Theory* **21** (3): 250–6. doi:10.1109/tit.1975.1055384.
3. ^ Bishop, M; Thompson E (20 July 1986). "Maximum likelihood alignment of DNA sequences". *J Mol Biol* **190** (2): 159–65. doi:10.1016/0022-2836(86)90289-5. PMID 3641921.
4. ^ Durbin, Richard (1998). *Biological Sequence Analysis*. Cambridge, UK: Cambridge University Press.
5. ^ Bilmes, Jeff A. (1998). *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*. Berkeley, CA: International Computer Science Institute. pp. 7–13.
6. ^ "Baum-Welch and HMM applications" (PDF). Johns Hopkins Bloomberg School of Public Health. Retrieved 2 October 2013.
7. ^ Frazzoli, Emilio. "Intro to Hidden Markov Models the Baum-Welch Algorithm" (PDF). Aeronautics and Astronautics, Massachusetts Institute of Technology. Retrieved 2 October 2013.
8. ^ Baker, J. (1975). "The DRAGON system—An overview". *IEEE Transactions on Acoustics, Speech, and Signal Processing* **23**: 24–29. doi:10.1109/TASSP.1975.1162650.
9. ^ Rabiner, Lawrence (Feb 1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". *Proceedings of the IEEE* **77** (2): 257–286. doi:10.1109/5.18626.
10. ^ Tokuda, Keiichi; Takayoshi Yoshimura; Takashi Masuko; Takao Kobayashi; Tadashi Kitamura (2000). "SPEECH PARAMETER GENERATION ALGORITHMS FOR HMM-BASED SPEECH SYNTHESIS". *IEEE International Conference on Acoustics, Speech, and*

Signal Processing **3**.
11. ^ Dingel, Janis; Joachim Hagenauer (24 June 2007). "Parameter Estimation of a Convolutional Encoder from Noisy Observations". *IEEE International Symposium on Information Theory*.
12. ^ Wright, Charles; Ballard, Lucas; Coull, Scott; Monrose, Fabian; Masson, Gerald (2008). "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations". *IEEE International Symposium on Security and Privacy*.
13. ^ Brumley, Bob; Hakala, Risto (2009). "Cache-Timing Template Attacks". *Advances in Cryptography* **5912**: 667–684. doi:10.1007/978-3-642-10366-7_39. Retrieved 21 October 2013.
14. ^ Salzberg, Steven; Arthur L. Delcher; Simon Kasif; Owen White (1998). "Microbial gene identification using interpolated Markov Models". *Nucleic Acids Research* **26** (2): 544–548. doi:10.1093/nar/26.2.544. PMC 147303. PMID 9421513.
15. ^ "Glimmer: Microbial Gene-Finding System". Johns Hopkins University - Center for Computational Biology.
16. ^ Delcher, Arthur; Kirsten A. Bratke; Edwin C. Powers; Steven L. Salzberg (2007). "Identifying bacterial genes and endosymbiont DNA with Glimmer". *Bioinformatics* **23** (6): 673–679. doi:10.1093/bioinformatics/btm009. PMC 2387122. PMID 17237039.
17. ^ Burge, Christopher. "The GENSCAN Web Server at MIT". Retrieved 2 October 2013.
18. ^ Burge, Chris; Samuel Karlin (1997). "Prediction of Complete Gene Structures in Human Genomic DNA". *J. Mol. Bio.* **268** (1): 78–94. doi:10.1006/jmbi.1997.0951. PMID 9149143.
19. ^ Burge, Christopher; Samuel Karlin (1998). "Finding the Genes in Genomic DNA". *Current Opinion in Structural Biology* **8**: 346–354. doi:10.1016/s0959-440x(98)80069-9.
20. ^ Korbel, Jan; Alexander Urban; Fabien Grubert; Jiang Du; Thomas Royce; Peter Starr; Guoneng Zhong; Beverly Emanuel; Sherman Weissman; Michael Snyder; Marg Gerstein (12 June 2007). "Systematic prediction and validation of breakpoints associated with copy-number variations in the human genome". *PNAS* **104** (24): 10110–5. doi:10.1073/pnas.0703834104.

External links [edit]

- A comprehensive review of HMM methods and software in bioinformatics - Profile Hidden Markov Models
- Early HMM publications by Baum:
  - A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains
  - An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology
  - Statistical Inference for Probabilistic Functions of Finite State Markov Chains
- The Shannon Lecture by Welch, which speaks to how the algorithm can be implemented efficiently:
  - Hidden Markov Models and the Baum–Welch Algorithm, IEEE Information Theory Society Newsletter, Dec. 2003.
- An alternative to the Baum–Welch algorithm, the Viterbi Path Counting algorithm:
  - R. I. A. Davis, B. C. Lovell, "Comparing and evaluating HMM ensemble training algorithms using train and test and condition number criteria", Pattern Analysis and Applications, vol. 6, no. 4, pp. 327–336, 2003.
- An Interactive Spreadsheet for Teaching the Forward-Backward Algorithm (spreadsheet and article with step-by-step walkthrough)
- Formal derivation of the Baum–Welch algorithm
- Implementation of the Baum–Welch algorithm

Categories: Statistical algorithms | Bioinformatics algorithms | Markov models