Return to Main

Objectives

Review:
Elements

Calculations:
HMM Evaluation
The Forward Algorithm
The Viterbi Algorithm
The EM Algorithm

On-Line Resources:
Parameter Estimation
Baum Welch
Expectation Maximization (EM)

LECTURE 23: PARAMETER ESTIMATION

- Objectives:
  - Review the components of a hidden Markov model
  - Forward probability calculation
  - The Viterbi algorithm
  - The Expectation Maximization (EM) algorithm
  - Understand the use of these techniques in HMM training and decoding

This lecture combines material from the course textbook:

> X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

> J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

ECE 8463: FUNDAMENTALS OF SPEECH RECOGNITION

Professor Joseph Picone
Department of Electrical and Computer Engineering
Mississippi State University

email: picone@isip.msstate.edu
phone/fax: 601-325-3149; office: 413 Simrall
URL: http://www.isip.msstate.edu/resources/courses/ece_8463

Modern speech understanding systems merge interdisciplinary technologies from Signal Processing, Pattern Recognition, Natural Language, and Linguistics into a unified statistical framework. These systems, which have applications in a wide range of signal processing problems, represent a revolution in Digital Signal Processing (DSP). Once a field dominated by vector-oriented processors and linear algebra-based mathematics, the current generation of DSP-based systems rely on sophisticated statistical models implemented using a complex software paradigm. Such systems are now capable of understanding continuous speech input for vocabularies of hundreds of thousands of words in operational environments.

In this course, we will explore the core components of modern statistically-based speech recognition systems. We will view speech recognition problem in terms of three tasks: signal modeling, network searching, and language understanding. We will conclude our discussion with an overview of state-of-the-art systems, and a review of available resources to support further research and technology development.

Tar files containing a compilation of all the notes are available. However, these files are large and will require a substantial amount of time to download. A tar file of the html version of the notes is available here. These were generated using wget:

```
wget -np -k -m
http://www.isip.msstate.edu/publications/courses/ece_8463/lectures/current
```

A pdf file containing the entire set of lecture notes is available here. These were generated using Adobe Acrobat.

Questions or comments about the material presented here can be directed to help@isip.msstate.edu.

(html, pdf)

17: Spectral Normalization
(html, pdf)

Parameterization:

18: Differentiation
(html, pdf)

19: Principal Components
(html, pdf)

20: Linear Discriminant Analysis
(html, pdf)

Acoustic Modeling:

21: Dynamic Programming
(html, pdf)

22: Markov Models
(html, pdf)

23: Parameter Estimation
(html, pdf)

24: HMM Training
(html, pdf)

25: Continuous Mixtures
(html, pdf)

26: Practical Issues
(html, pdf)

27: Decision Trees
(html, pdf)

28: Limitations of HMMs
(html, pdf)

Language Modeling:

LECTURE 23: PARAMETER ESTIMATION

- Objectives:
  - Review the components of a hidden Markov model
  - Forward probability calculation
  - The Viterbi algorithm
  - The Expectation Maximization (EM) algorithm

- Understand the use of these techniques in HMM training and decoding

This lecture combines material from the course textbook:

> X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

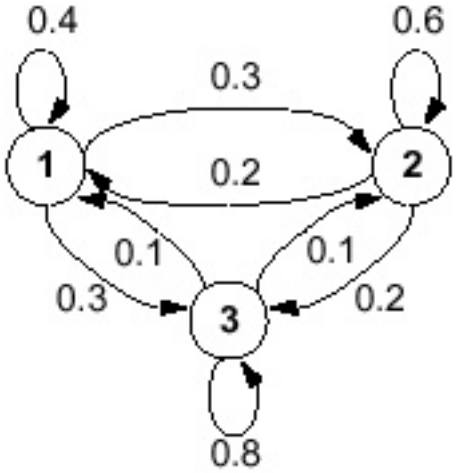
> J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

ELEMENTS OF AN HIDDEN MARKOV MODEL

Recall the elements of a hidden Markov Model:

- N — the number of states
- M — the number of distinct observations per state
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(k)\}$
- The initial state distribution $\pi = \{\pi_i\}$

We can write this succinctly as: $\lambda = (\underline{A}, \underline{B}, \pi)$



There are three problems we must solve:

- The Evaluation Problem: Given a model and a set of observations, what was the probability that the model produced these observations?

- The Decoding Problem: What was the most likely state sequence in the model that produced a given set of observations?

- The Learning Problem: Given a model and a set of observations, how can we adjust the model parameters to increase the probability of the observations (data) given the model?

HMM EVALUATION (EXHAUSTIVE SEARCH)

Denote any partial sequence of observations in time by:

$$y_{t_1}^{t_2} \equiv \{y(t_1), y(t_1+1), y(t_1+2), \ldots, y(t_2)\}$$

The forward partial sequence of observations at time t is

$$y_1^t \equiv \{y(1), y(2), \ldots, y(t)\}$$

The backward partial sequence of observations at time t is

$$y_{t+1}^T \equiv \{y(t+1), y(t+2), \ldots, y(T)\}$$

A complete set of observations of length T is denoted as $y \equiv y_1^T$.

What is the likelihood of an HMM?

We would like to calculate $P(\lambda|\underline{y} = y)$ — however, as we will explain later, we can't. Instead, we can calculate $P(\underline{y} = y|\lambda)$. Consider the brute force method of computing this. Let $\vartheta = \{i_1, i_2, \ldots, i_T\}$ denote a specific state sequence. The probability of a given observation sequence being produced by this state sequence is:

$$P(y|\vartheta,\lambda) = b(y(1)|i_1)b(y(2)|i_2)\ldots b(y(T)|i_T)$$

The probability of the state sequence is

$$P(\vartheta|\lambda) = P(\underline{x}(1) = i_1)a(i_2|i_1)a(i_3|i_2)\ldots a(i_T|i_{T-1})$$

Therefore,

$$P(y,(\vartheta|\lambda)) = P(\underline{x}(1) = i_1)a(i_2|i_1)a(i_3|i_2)\ldots a(i_T|i_{T-1})$$
$$x\ \ b(y(1)|i_1)b(y(2)|i_2)\ldots b(y(T)|i_T)$$

To find $P(y|\lambda)$, we must sum over all possible paths:

$$P(y|\lambda) = \sum_{\forall\vartheta} P(y,(\vartheta|\lambda))$$

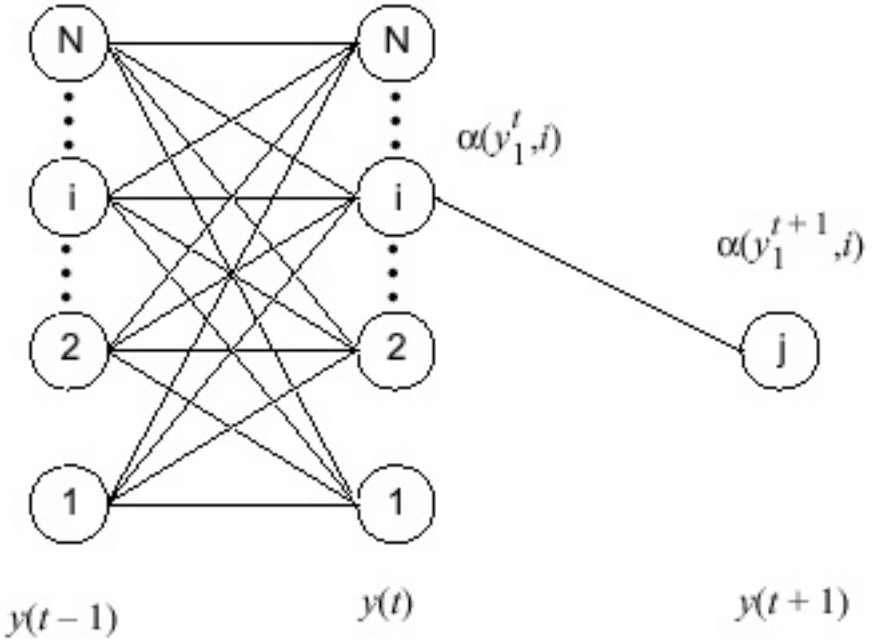
This requires $O(2TN^T)$ flops. For $N = 5$ and $T = 100$, this gives about 1.6×10^{72} computations per HMM!

THE FORWARD ALGORITHM

The *forward algorithm* begins by defining a "forward-going" partial probability sequence:

$$\alpha(y_1^t) \equiv P(y_1^t = y_1^t, \underline{x}(t) = i \,|\, \lambda)$$

Let us next consider the contribution to the overall sequence probability made by a single transition:



$$\begin{aligned}\alpha(y_1^{t+1}, j) &= \alpha(y_1^t, i) P(\underline{x}(t+1) = j \,|\, \underline{x}(t) = i) \times \\ &\quad P(\underline{y}(t+1) = y(t+1) \,|\, \underline{x}(t+1) = j) \\ &= \alpha(y_1^t, i) a(j|i) b(y(t+1)|j)\end{aligned}$$

Summing over all possibilities for reaching state "j":

$$\alpha(y_1^{t+1}, j) = \sum_{i=1}^{N} \alpha(y_1^t, i) a(j|i) b(y(t+1)|j)$$

The recursion is initiated by setting:

$$\alpha(y_1^t, j) = P(\underline{x}(1) = j) b(y(1)|j)$$

We still need to find $P(y|\lambda)$:

$$P(y, \underline{x}(t) = i|\lambda) = \alpha(y_1^t, i)\beta(y_{t+1}^T \Big| i)$$

for any state i. Therefore,

$$P(y|\lambda) = \sum_{i=1}^{N} \alpha(y_1^T, i)$$

These equations suggest a recursion in which, for each value of t we iterate over ALL states and update $\alpha(y_1^t, j)$. When $t = T$, $P(y|\lambda)$ is computed by summing over ALL states:

The Forward Algorithm

Step 1: Initialization

$$\alpha(y_1^1, j) = \pi_j b(y(1)|j) \qquad 1 \le j \le N$$

Step 2: Induction

$$\alpha(y_1^{t+1}, j) = \left[\sum_{i=1}^{N} \alpha(y_1^t, i) a(j|i) \right] b(y(t+1)|j)$$

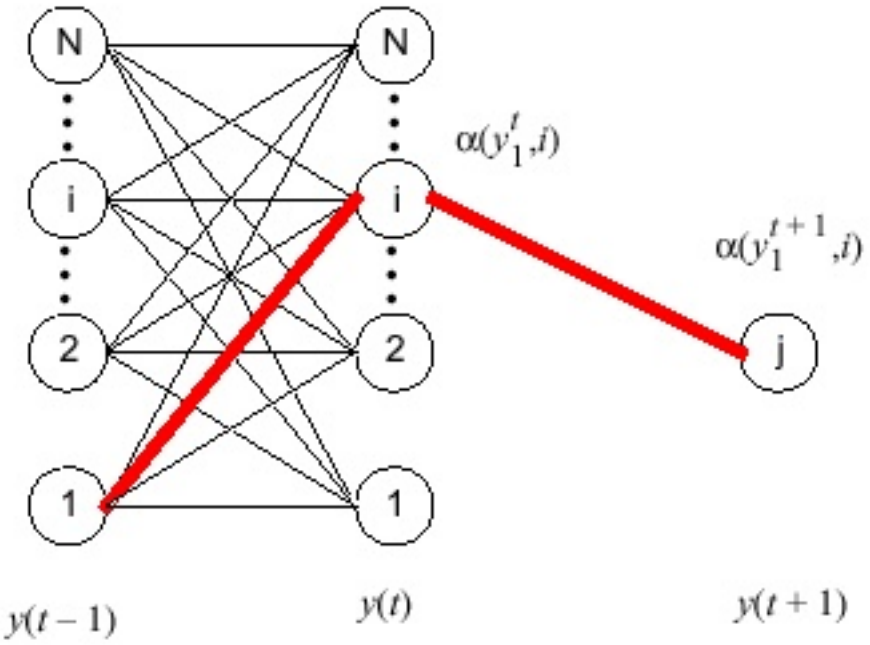
Step 3: Termination

$$P(y|\lambda) = \sum_{i=1}^{N} \alpha(y_1^T, i)$$

The complexity of this algorithm is $O(N^2 T)$, or for $N = 5$ and $T = 100$, approximately 2500 flops are required (compared to 10^{72} flops for the exhaustive search method).

THE VITERBI (BEST PATH) ALGORITHM

The *Viterbi algorithm* can also be used to find the best state sequence. Note that the principal difference is that we model the overall sequence probability by the probability of the single best path:



The Viterbi Algorithm

Step 1: Initialization

$$V_1(j) = \pi_j b(y(1)|j) \qquad 1 \le j \le N$$

$$B_1(j) = 0$$

Step 2: Induction

$$V_t(j) = \underset{1 \le i \le N}{Max}[V_{t-1}(i)a_{ij}]b(y(t)|j) \qquad 1 \le j \le N \quad 2 \le t \le T$$

$$B_t(j) = \underset{1 \le i \le N}{Argmax,}[V_{t-1}(i)a_{ij}]$$

Step 3: Termination

$$P(y|\lambda) = \underset{1 \le i \le N}{Max}[V_T(i)] \qquad s_t^* = \underset{1 \le i \le N}{Argmax,}[B_T(i)]$$

Step 4: Backtracking

$$s_t^* = B_{t+1}(s^*_{t+1}) \qquad S^* = (s_T^*, s^*_{T-1}, \ldots, s_1^*)$$

Note that the complexity for the Viterbi algorith is $O(N^2T)$. Often it is much less than this because the topology being scored is extremely restrictive (left-to-right models).

THE EXPECTATION MAXIMIZATION (EM) ALGORITHM

- The Expectation Maximization (EM) algorithm can be viewed as a generalization of maximum likelihood parameter estimation (MLE) when the data observed is incomplete. We observe some data y, and seek to maximize $P(Y = y|\lambda)$. However, in order to do this, we need to know some hidden data x.

- We assume a parameter vector λ and estimate the probability that each x occurred in the generation of y. In this way, we can assume we observed the pair (x, y) with probability $P(X = x, Y = y|\lambda)$.

- To compute the new $\bar{\lambda}$, we use the maximum likelihood estimate of λ.

- Does this process converge?

According to Bayes' rule:

$$P(X = x, Y = y|\bar{\lambda}) = P(X = x|Y = y, \bar{\lambda})P(Y = y|\bar{\lambda})$$

The log likelihood can be expressed as:

$$\log P(Y = y|\bar{\lambda}) = \log P(X = x, Y = y|\bar{\lambda}) - \log P(X = x|Y = y, \bar{\lambda})$$

We take the conditional expectation of $\log P(Y = y|\bar{\lambda})$ over X:

$$E_{\lambda}[\log P(Y = y|\bar{\lambda})]_{X|Y=y} = \sum_{x} (P(X = x|Y = y, \lambda)) \log P(Y = y|\bar{\lambda})$$

$$= \log P(Y = y|\bar{\lambda})$$

Combining the previous two expressions:

$$\log P(Y = y|\bar{\lambda}) = E_{\lambda}[\log P(X, Y = y|\bar{\lambda})]_{X|Y=y} - E_{\lambda}[\log P(X|Y = y, \bar{\lambda})]_{X|Y=y}$$

$$= Q(\lambda, \bar{\lambda}) - H(\lambda, \bar{\lambda})$$

The convergence of the EM algorithm lies in the fact that if we choose $\bar{\lambda}$ such that $Q(\lambda, \bar{\lambda}) \geq Q(\lambda, \lambda)$, then $\log P(Y = y|\bar{\lambda}) \geq \log P(Y = y|\lambda)$.

This follows because we can show that $H(\lambda, \bar{\lambda}) \leq H(\lambda, \lambda)$ using a special case

of Jensen's inequality ($\sum_x p(x)\log p(x) \geq \sum_x p(x)\log q(x)$).

A summary of the procedure is:

The EM Algorithm

Step 1: Choose an initial estimate λ.

Step 2: **E-step**: Compute auxiliary Q-function $Q(\lambda, \bar{\lambda})$ (which is also the expectation of the log likelihood of the data) based on λ.

Step 3: **M-step**: Compute $\hat{\lambda} = \arg\max Q(\lambda, \bar{\lambda})$ to maximize the auxiliary Q-function.

Step 4: Iteration: Set $\lambda = \bar{\lambda}$, and repeat from Step 2 until convergence.

Next: The Viterbi Algorithm **Up:** Parametrization: Choice of Distributions **Previous:** Continuous Emissions

Parameter Estimation

For all the above distributions, the EM (Expectation-Maximization) algorithm [&make_named_href(", "node31.html#Dempster77","[52]"), &make_named_href(", "node31.html#Baum67","[16]"), &make_named_href(", "node31.html#Baum70","[17]"), &make_named_href(", "node31.html#Baum72","[18]")] can be used to estimate the parameters of the HMM in order to **maximize the likelihood function** $l(\theta) = P_\theta(\mathcal{D}) = \prod_p P_\theta(y_1^{T_p}(p))$ over the set $\mathcal{D}$ of training sequences (indexed by the letter *p*). The EM algorithm itself is discussed in section 5.2. In this application of EM, the state variable Q_t is viewed as the missing variable. It was shown early on [&make_named_href(", "node31.html#Baum66","[53]")] that the maximum likelihood estimates of the parameters of a hidden Markov process are consistent. More recently, it has been shown [&make_named_href(", "node31.html#bickel+ritov95","[54]")] that for HMMs, maximum likelihood estimates are asymptotically normal as expected and consistent estimates of their variance can be constructed, so that the estimation procedure is asymptotically valid.

Other emission distributions of the exponential family (or mixtures thereof) could also be used. In speech and other sequence recognition applications, the EM algorithm can also be used when the state sequence is constrained, which corresponds to modeling the conditional distribution $P(y_1^T | w_1^L)$, e.g., with $w_1^L = \{w_1, \ldots, w_L\}$ a sequence of ``correct" labels which should be associated with the observed sequence $Y_1^T = y_1^T$. When the states are associated to labels, conditioning on w_1^L corresponds to a restriction on the graph representing topology of the HMM: see for example how the unconstrained recognition model in Figure 4 is constrained with the correct word sequence ``the dog" in Figure 8.

It should be noted that other criteria than the maximum likelihood criterion can be used to train HMMs, for example to incorporate a prior on parameters, to make the training more discriminant (focus more on doing the classification correctly), or to make it more robust (penalized maximum likelihood [&make_named_href(", "node31.html#Silverman-encyc86","[55]")]). For more complex distributions than those described above or for several learning criteria other than maximum likelihood, numerical optimization methods other than the EM algorithm are often used, usually based on the gradient of the learning criterion with respect to the free numerical parameters. When possible, the EM algorithm is generally preferred because of its faster convergence properties. These topics will be further discussed in sections 5.2 and 3.5.

Next: The Viterbi Algorithm **Up:** Parametrization: Choice of Distributions **Previous:** Continuous Emissions

Yoshua Bengio
Tue Oct 7 08:34:36 EDT 1997

Parameter Estimation for HMMs

In examples 6.1.2a [*] and 6.1.2b [*] we constructed hidden Markov models knowing the transition and emission probabilities for the problems we had to solve. In real life, this may not be the case. We may be given *n* strings $X^{(1)},\ldots,X^{(n)} \in \Sigma^*$ of length $L^{(1)},\ldots,L^{(n)}$, respectively, which were all generated from the HMM $\mathcal{M} = (\Sigma, Q, \Theta)$. The values of the probabilities in Θ, however, are unknown a-priori.

In order to construct the HMM that will best characterize $X^{(1)},\ldots,X^{(n)}$, we will have to assign values to Θ that will maximize the probabilities of our strings according to the model. Since all strings are assumed to be generated independently, we can write:

$$P(X^{(1)},\ldots,X^{(n)}|\Theta) = \prod_{j=1}^{n} P(X^{(i)}|\Theta) \tag{32}$$

Using the logarithmic score, our goal is to find Θ^* such that

$$\Theta^* = \arg\max_{\Theta}\{Score(X^{(1)},\ldots,X^{(n)}|\Theta)\} \tag{33}$$

where:

$$Score(X^{(1)},\ldots,X^{(n)}|\Theta) = \log P(X^{(1)},\ldots,X^{(n)}|\Theta) = \sum_{j=1}^{n} \log(P(X^{(i)}|\Theta)) \tag{34}$$

The strings $X^{(1)},\ldots,X^{(n)}$ are usually called the *training sequences*.

Case 1: Assuming we know the state sequences $\Pi^{(1)}, \ldots, \Pi^{(n)}$ corresponding to $X^{(1)}, \ldots, X^{(n)}$, respectively. We can scan these sequences and compute:

- A_{kl} - the number of transitions from the state k to l.
- $E_k(b)$ - the number of times that an emission of the symbol b occurred in state k.

The maximum likelihood estimators will be:

$$a_{kl} = \frac{A_{kl}}{\sum_{q \in Q} A_{kq}} \tag{35}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{\sigma \in \Sigma} E_k(\sigma)} \tag{36}$$

To avoid zero probabilities, when working with a small amount of samples, it is recommended to work with A'_{kl} and $E'_k(b)$, where:

$$A'_{kl} = A_{kl} + r_{kl} \tag{37}$$
$$E'_k(b) = E_k(b) + r_k(b) \tag{38}$$

Usually the *Laplace correction*, where all r_{kl} and $r_k(b)$ values equal 1, is applied, having an intuitive interpretation of a-priori assumed uniform distribution. However, it may be beneficial in some cases to use other values for the correction (e.g. when having some prior information about the transition or emission probabilities).

Case 2: Usually, the state sequences $\Pi^{(1)}, \ldots, \Pi^{(n)}$ are not known. In this case, the problem of finding the optimal set of parameters Θ^* is known to be NP-complete. The *Baum-Welch algorithm* [2], which is a special case of the *EM technique (Expectation and Maximization)*, can be used for heuristically finding a solution to the problem.

1. *Initialization*: Assign values to Θ.
2. *Expectation*:
   (a) Compute the expected number of state transitions from state k to state l. Using the same

arguments we used for computing $P(X, \pi_i = k)$ (see 6.30), we get:

$$P(\pi_i = k, \pi_{i+1} = l | X, \Theta) = \frac{f_k(i) \cdot a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1)}{P(X)} \quad (39)$$

Hence, we can compute the expectation:

Let $\{f_k^{(j)}(i), b_k^{(j)}(i)\}_{k \in Q, \; i \leq L(j)}$ denote the forward and backward probabilities of the string $X^{(j)}$. Then

$$A_{kl} = \sum_{j=1}^{n} \frac{1}{P(X^{(j)})} \cdot \sum_{i=1}^{L(j)} f_k^{(j)}(i) \cdot a_{kl} \cdot e_l(x_{i+1}^{(j)}) \cdot b_l^{(j)}(i+1) \quad (40)$$

(b)

Compute the expected number of emissions of the symbol *b* that occurred at the state *k* (using the value of $P(\pi_i = k | X)$ as calculated in 6.31):

$$E_k(b) = \sum_{j=1}^{n} \frac{1}{P(X^{(j)})} \cdot \sum_{\{i | x_i^{(j)} = b\}} f_k^{(j)}(i) \cdot b_k^{(j)}(i) \quad (41)$$

3.

Maximization: Re-compute the new values for Θ from A_{kl} and $E_k(b)$, as explained above (in case 1).

4.

Repeat steps 2 and 3 until the improvement of $Score(X^{(1)}, \ldots, X^{(n)} | \Theta)$ is less then a given parameter ϵ.

The EM-algorithm guarantees that the target function values $Score(X^{(1)}, \ldots, X^{(n)} | \Theta)$ are monotonically increasing, and as logarithms of probabilities are certainly bounded by 0, the algorithm is guaranteed to converge. It is important to notice that the convergence is of the target function and not in the Θ space: the values of Θ may change drastically even for almost equal values of the target function, which may imply that the obtained solution is not stable.

The main problem with the Baum-Welch algorithm is that there may exist several local maxima of the target function and it is not guaranteed that we reach the global maximum: the convergence may lead to a local maximum. A useful way to circumvent this pitfall is to run the algorithm several times, each time with different initial values for θ. If we reach the same maximum most of the times, it is highly probable that this is indeed the global maximum. Another way is to start with θ values that are meaningful, like in the case of the CpG islands we might start from θ values obtained from a real case statistics.

---

next up previous

Next: Profile Alignment **Up:** Hidden Markov Models **Previous:** Forward and Backward Probabilities

Peer Itsik
2000-12-19

EKSL Digital Library

EKSL : Statistics : Expectation-Maximization

Contents (3 total):

- **A Gentle Tutorial on the EM algorithm including Gaussian Mixtures and Baum-Welch**
  Jeff Bilmes
  UC Berkely Technical Report 1997
  MOAB: Bilmes-gentle
  /nfs/genet/users1/eksl/library/statistics/expectation-maximization/EM.Tutorial.pdf

  We describe the maximum-likelihood parameter estimation problem and how the Expectation-Maximization (EM) algorithm can be used for its solution. We first describe the abstract form of the EM algorithm as it is often given in the literature. We then develop the EM pa-rameter estimation procedure for two applications: 1) finding the parameters of a mixture of Gaussian densities, and 2) finding the parameters of a hidden Markov model (HMM) (i.e., the Baum-Welch algorithm) for both discrete and Gaussian mixture observation models. We derive the update equations in fairly explicit detail but we do not prove any conver-gence properties. We try to emphasize intuition rather than mathematical rigor.

- **The EM Algorithm**
  Stuart Russell
  Unpublished
  MOAB: Russell-EM
  /nfs/genet/users1/eksl/library/statistics/expectation-maximization/Russel.EM.pdf

  This document is to derive the algorithm in its most general form from first principles and to give a short proof of its convergence. The derivation extends the mixture-model derivation from Bishop (1995, pp. 65--66) and leads to the algorithm given in Mitchell (1997, p.195). Suppose we define the log likelihood function L(`) = ln P (Xj`) and suppose that our current estimate for the optimal parameters is ` i . We will examine what happens to L when a new value ` is computed by the algorithm: L(`) \Gamma L(` i) = ln P (Xj`) \Gamma ln P (Xj` i) = ln

- **Maximum Likelihood from Incomplete Data via the EM Algorithm**
  A. P. Dempster, N. M. Laird, D. B. Rubin
  Journal of the Royal Statistical Society. Series B (Methodological), Volume 39, Issue 1 (1977),

1-38.
MOAB: Dempster77
/nfs/genet/users1/eksl/library/statistics/expectation-maximization/dempster.pdf

This paper explains the EM algorithm at a very broad level.

EKSL : Statistics : Expectation-Maximization

Site Map
Last update: 1/23/2002